Se former autrement

HONORIS UNITED UNIVERSITIES

2022 / 2023

SPECIALTY: Data Science

Enhancing the Recruitment Process with a Data-Driven Solution

Produced by: Wissem Ellefi

ESPRIT Supervisor: Aymen Esselmi

Company Supervisor: Souleima Zghab

I validate the submission of the student's report:

- Name & Surname : Wissem Ellefi

Business site Supervisor

- Name & Surname : Souleima Zghab

Stamp & Signature

TALAN TUNISIE CONSULTING
Rue de l'Energie Solaire Impasse 1
2035 Charguia 1
Tél: 70.015.010
Fax: 70.015.007
M.F : 1325392 R/A/M/000

[illegible]

Academic Supervisor

- Name & Surname : Aymen Esselmi

Signature

[illegible]

Data Science Thesis

A thesis submitted in partial fulfillment of the requirements
for the degree of Engineering

by

Wissem Ellefi

ESPRIT University: The Private Higher School of Engineering and Technology

Year 2023

Abstract

In the realm of the recruitment industry, the prevailing methodologies exhibit limitations in effectively managing extensive datasets. These approaches necessitate manual intervention for the screening and assessment of resumes, resulting in a protracted and inefficient procedure. As the volume of candidates escalates, the recruitment team grapples with the challenge of retaining comprehensive insights into each individual applicant. Conversely, job seekers invest substantial time in identifying employment opportunities that align with their profiles.

This thesis illuminates the transformative potential of artificial intelligence in mitigating these challenges. By harnessing the power of AI, the time investment required for evaluating a candidate's profile by HR personnel can be significantly curtailed from an average of approximately 7 minutes to a task seamlessly executed by artificial intelligence. This transition not only expedites the evaluation process but also streamlines it into an automated workflow.

Moreover, this research delves into the augmentation of job seekers' experiences through the implementation of recommendation and matching algorithms.
By leveraging these advanced algorithms, the likelihood of identifying fitting job opportunities is substantially enhanced. Consequently, job seekers are empowered to navigate the employment landscape more efficiently and with a heightened probability of securing positions that harmonize with their qualifications and aspirations.

In summation, this thesis demonstrates the efficacy of integrating artificial intelligence into the recruitment domain to revolutionize candidate evaluation, optimize resource utilization, and amplify the efficacy of job matching for enhanced outcomes.

Acknowledgements

I express my heartfelt gratitude to my parents, whose unwavering love, support, and encouragement have been the driving force behind my academic and personal journey. Their belief in my abilities and constant encouragement has been a constant source of motivation, and I am truly grateful for their sacrifices and dedication.

I am indebted to all the mentors, professors, and educators who have played instrumental roles in shaping my knowledge and skills throughout my academic pursuit. Their guidance, wisdom, and passion for teaching have inspired me to strive for excellence and pursue my academic interests with determination.

I extend my thanks to my friends and colleagues who have been there for me through every challenge and triumph. Your support and camaraderie have made this journey more enjoyable and rewarding.

I am also grateful to all the individuals who have believed in me, even through a simple word of encouragement or a kind gesture. Your belief in my potential has bolstered my confidence and strengthened my resolve to reach for my goals.

Lastly, I acknowledge the invaluable contribution of the wider academic community, whose collective efforts and groundbreaking research have paved the way for the development of the field I am so passionate about.

Introduction

IN today's rapidly evolving job market, the process of connecting job seekers with suitable job opportunities and assisting recruiters in identifying top-quality candidates has become increasingly complex. Therefore, traditional approaches to recruitment often fall short, resulting in time-consuming processes, mismatches, and limited visibility into applicant qualifications. In response to these pressing challenges, this thesis presents an innovative solution that leverages advanced recommender systems and artificial intelligence (AI) algorithms using Python to transform the way recruiters and job seekers connect. Each chapter of this thesis encapsulates a sprint-focused endeavor aimed at uncovering and addressing specific aspects of the recruitment landscape. Through a systematic breakdown of the recruitment process, we delve into five key dimensions within each sprint.
The "Background and Context" section serves as the foundational backdrop, furnishing essential information and contextual nuances that establish the platform for the ensuing topic's exploration. In "Implementation and Execution," a comprehensive dive into the project's realization and operationalization is undertaken, intricately dissecting the undertaken steps and employed methodologies. The domain of "Results and Findings" unfolds to unveil the consequential yields and discoveries borne out of the project's materialization, accentuating noteworthy insights garnered along the way. The segment entitled "Lessons Learned and Reflection" encapsulates the chapter's educative takeaways, intertwining acquired wisdom with an introspective examination of surmounted obstacles and their remedies. Ultimately, in the "Conclusion and Next Steps," the chapter's essence is distilled, underscored by its importance, and culminates in outlining potential trajectories forward a roadmap shaped by the achieved outcomes.
This structured approach will ensure that the reader gains a comprehensive understanding of the project from its inception to its outcomes and potential future developments.

Keywords: recommender system, Data-driven, Artificial Intelligence, Dashboard, Optimize, Recruitement, Analytics, candidate Selection.

Contents

List of Figures

List of Tables

Chapter 1

Project Overview

Introduction

In this crucial chapter, we delve into the intricacies of project implementation and planning. Here, we outline the key elements that contribute to the successful execution of our project, including the project methodology, timeline, objectives, requirements, and employed technologies.

1.1 Presentation of the Project

In this section, we present an in-depth exploration of our project, highlighting its objectives, problem statement, requirements, Background and project diagrams.

1.1.1 Background and Significance of the Project

Within this pivotal section, we delve into the academic background, the Host Company and the profound significance of our project.

Academic Background

This project draws upon the strong academic background and foundation provided by the Engineering program at ESPRIT University. The rigorous curriculum and comprehensive coursework have equipped me with the necessary knowledge and skills to undertake this project successfully.

Host Company and Professional Background

Talan, our esteemed host company, is a prominent French consulting firm that has garnered a reputation for its expertise in technological innovation and transformation. With a rich history spanning over 15 years, Talan has consistently demonstrated its prowess in advising, supporting, and successfully executing transformational projects for a diverse clientele comprising both companies and governmental bodies. Operating across a global spectrum, Talan has established its presence in several countries, including France, Belgium, Canada, Spain, Luxembourg, the United Kingdom, United States, Switzerland and Tunisia. Their extensive geographical reach underscores their commitment to providing comprehensive solutions and innovative strategies for organizations seeking to thrive in an ever-evolving technological landscape. Talan's dedication to excellence and its global presence makes it a formidable partner for enterprises embarking on transformative journeys.

Figure 1.1: Talan's Logo

The Importance of the Project

The importance of this project holds significant weight in my pursuit of attaining my engineering degree. As part of my academic requirements, a six-month internship holds a crucial place in my curriculum, serving as a bridge between theoretical knowledge and practical application. This project serves as a cornerstone in my journey, providing me with an opportunity to gain hands-on experience, apply the skills and concepts learned throughout my engineering program, and demonstrate my ability to tackle real-world challenges.

1.1.2 Problem Statement and Motivation

Delving into the problem statement our project aims to address, we offer a clear and concise description of the challenges faced by stakeholders in the recruitment industry. By emphasizing pain points and limitations, we establish the context and significance of our project, followed by highlighting the project's motivation.

Problem Statement

In today's dynamic job market, both recruiters and job seekers encounter a myriad of obstacles when it comes to establishing meaningful connections between them. The conventional methods of recruitment heavily depend on manual screening procedures, which often prove to be laborious and time-consuming, leading to some inefficiencies. On the first hand, Recruiters find themselves burdened with the arduous task of meticulously reviewing countless candidates' resumes and applications, particularly as the volume of job seekers continues to rise. Consequently, it becomes increasingly difficult for them to thoroughly assess each application. On the other hand, job seekers face their own set of challenges, struggling to identify suitable employment opportunities that align with their qualifications and preferences. Many job openings fail to receive effective advertising or reach a broad audience which results in missed chances and opportunities for both job seekers and recruiters alike.

Motivation

My motivation to undertake this project stems from the desire to address the problem and make a meaningful contribution to the recruitment and job-matching domain.
Also, undertaking this project provides me with a platform for personal and professional growth. By taking on the challenges associated with developing a comprehensive solution, I expand my knowledge, skills, and expertise in the data science field.

1.1.3 Objectives

The main objective of this project is to develop a data-driven solution that provides personalized job recommendations to job seekers and assists recruiters in finding the most suitable candidates for their job openings. By applying AI algorithms and advanced recommender systems, the project aims to streamline the job search experience for job seekers and enable recruiters to access a pool of top-quality candidates. Therefore, we can categorize the goals into two distinct classes, namely, specific objectives for the recruiter and separate ones for the job seeker. By segregating these goals, we gain a comprehensive understanding of the distinct motivations driving each party involved in the recruitment process .

Objectives for Recruiters

For the recruiter, the objectives primarily revolve around finding the most qualified candidates to fill a vacant position.
Therefore, here are the objectives fixed for recruiters:

- **Streamline candidate selection:** Develop an AI-powered system that automates and streamlines the candidate selection process, allowing recruiters to efficiently identify the most suitable candidates for their job openings.
- **Enhance candidate evaluation:** Provide detailed reports and profiles for each applicant, including qualifications, skills, and performance indicators, to help recruiters make informed decisions during the hiring process and effectively evaluate candidates.
- **Save time and effort:** By leveraging the candidate's selection approach instead of doing that manually, reduce the time and effort required for manual screening of resumes and applications, allowing recruiters to focus on evaluating the most promising candidates.

Objectives for Job-seekers

On the other hand, the job seeker's objectives focus on securing gainful employment that aligns with their professional aspirations and personal goals.

Hereby, here are the objectives fixed for job seekers:

- **Personalized job recommendations:** Provide personalized job recommendations based on job seekers' profiles, skills, and preferences, enhancing the job search experience, and increasing the likelihood of finding the right job that aligns with their qualifications.
- **Boost job satisfaction:** By matching job seekers with relevant job opportunities that align with their qualifications and preferences, increase job satisfaction levels, and enhance the overall job search experience.

Interpretation

Thus, by recognizing and acknowledging the diverse goals of recruiters and job seekers, we can develop targeted strategies and approaches to facilitate successful matches between candidates and job opportunities.

1.1.4 Analysis of the Existing Solutions

To optimise its recruitment procedures, Talan currently relies on a third-party recruitment platform, namely SmartRecruiter. SmartRecruiter, as a widely recognized and established recruitment solution, undoubtedly offers a range of features and functionalities tailored to streamline the hiring process. However, as Talan's recruitment needs have evolved and matured, certain limitations and considerations have arisen that necessitate a reevaluation of their approach.

With its user-friendly interface and centralized management features, Smart Recruiters simplifies many aspects of the recruitment process. Nevertheless, it is essential to acknowledge that SmartRecruiter's advantages come at a significant financial cost. The licensing fees associated with the platform can be substantial, making it a substantial financial commitment for organizations like Talan.

1.1.5 Functional Requirements

These requirements are thoughtfully crafted to enhance the recruitment process for job seekers and recruiters, resulting in better hiring outcomes and increased job satisfaction.
Hence, my solution should be able to perform:

- Job Matching and Recommendation:
  - The system should employ advanced recommender systems and AI algorithms to match job seekers with suitable job openings.
  - The recommendation engine should consider factors such as qualifications, skills, experience, and salary of the job seekers.

- Reporting and Analysis:
  - Recruiters should be able to access statistical data and insights to support decision-making during the hiring process.
  - The system should generate reports and dashboards on job seekers, including their qualifications, experience, and Key performance indicators.
- Candidate Profile Analysis:
  - The system should use natural language processing (NLP) techniques to understand and extract information from candidates' resumes.
- Candidate Qualification Matching:
  - The system should be able to compare the qualifications and attributes of candidates against the job description specified by the recruiter.
  - The System should use algorithms and matching techniques to determine the suitability and compatibility of candidates for a particular job.
  - The system should assign a ranking or score to each candidate, indicating their level of qualification and suitability for the position.
  - The system should assign a ranking or score to each candidate, indicating their level of qualification and suitability for the position.

1.1.6 Non-Functional Requirements

In addition to the functional requirements that define the features and capabilities of a system, non-functional requirements play a crucial role in shaping the overall performance, reliability, and usability of the solution. Non-functional requirements are the quality attributes that specify how the system should behave and perform, rather than what it should do.
Therefore, the following are the key non-functional requirements that the solution should address, in order to ensure optimal performance:

- Performance:
  - To ensure optimal performance, the system leverages distributed computing using the Python Spark API (PySpark). By harnessing the power of distributed processing, the system can handle large volumes of data efficiently, enabling faster processing and analysis of candidate profiles.
  - Furthermore, the solution utilizes Sparse Arrays as a data structure for recommendation computations. Sparse Arrays are particularly beneficial in scenarios where data contains numerous zeros, as is often the case in candidate profiles.
- Accuracy:

  - To ensure high accuracy, the candidate analysis and recruitment system utilizes a diverse range of metrics such as accuracy, hit rate, cumulative hit rate, average reciprocal hit rank, coverage, diversity, and novelty. These metrics enable comprehensive evaluation and comparison of different approaches and techniques, ensuring precise candidate matching and reliable job recommendations.

- Maintainability:
  To ensure efficient management and future enhancements. Several techniques have been employed to facilitate maintainability throughout the development process:

  - Object-Oriented Programming (OOP) principles have been utilized to structure the project. By organizing code into classes and modules, the system promotes modularity and encapsulation, making it easier to understand, modify, and extend the functionality. ‘
  - A well-established and comprehensive project structure has been implemented, adhering to best practices and industry standards.
  - Use of non-code elements, such as comments, has been incorporated. Clear and descriptive comments are strategically placed throughout the codebase, explaining the purpose, functionality, and rationale behind critical sections. All parameters and settings are externalized, allowing them to be easily modified without altering the source code. This approach ensures that system behavior can be adjusted without the need for extensive code changes.

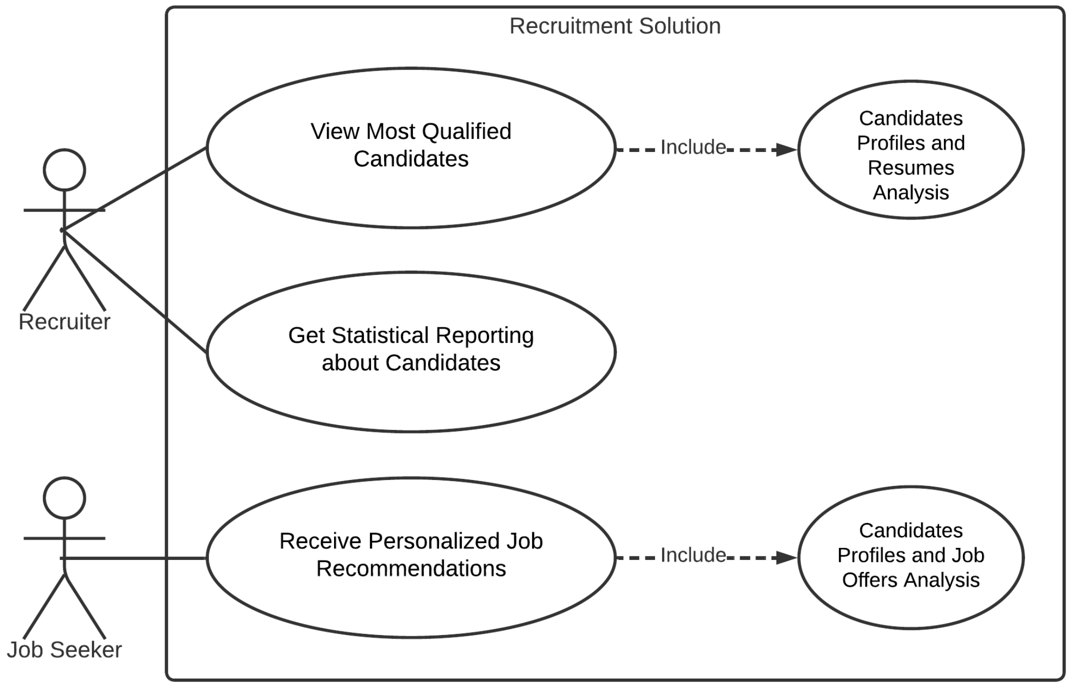
- Scalability:

  - To accommodate growing data volumes and increasing user demands. The system leverages the Spark distributed system. The distributed nature of Spark also allows the system to scale seamlessly with an increasing number of users.

1.1.7 Use Case Diagram

In the context of the recruitment system, the use case diagram illustrates the various actions that recruiters and job seekers can perform within the system.

The following diagram illustrates the primary actions and interactions within the recruitment system, specifically focusing on a data science project’s perspective.



Recruitment Solution
View Most Qualified Candidates
Include
Candidates Profiles and Resumes Analysis
Recruiter
Get Statistical Reporting about Candidates
Receive Personalized Job Recommendations
Include
Candidates Profiles and Job Offers Analysis
Job Seeker

Figure 1.2: Use Case Diagram

In our use case diagram, there are two primary actors: the ”Recruiter” and the ”Job Seeker.” The ”Recruiter” actor is capable of performing two main use cases. The first one is ”View most qualified candidates,” which involves the analysis of ”candidates’ profiles and resumes.” On the other hand, the ”Job Seeker” actor can execute the ”Receive personalized job recommendations” use case, which entails the analysis of candidates’ profiles and job offers.

Interpretation

These use cases represent the primary interactions and functionalities available to both the recruiter and the job seeker within the recruitment system.

1.2 Comparative Analysis and Selection of Work Methodology

Our subsequent sections focus on pinpointing the optimal methodology aligning with our project goals for efficiency and successful execution.

1.2.1 Agile Methodology

The methodology chosen for the entire project is the Agile methodology[4]. This approach emphasizes iterative and collaborative development, enabling teams to adapt to changing requirements and deliver incremental value throughout the project lifecycle. The decision to adopt

Agile Scrum was influenced by the existing practices and preferences of the team I joined, who are experienced and well-versed in Agile methodologies.
Some of the benefits of the Agile Scrum methodology include:

- **Flexibility and Adaptability:** Agile allows for flexibility in responding to evolving project requirements, feedback, and changing priorities. It enables teams to adapt their plans, prioritize work items, and make adjustments based on emerging insights, ultimately ensuring that the project remains aligned with stakeholder expectations.
- **Continuous Delivery of Value:** Agile promotes the delivery of incremental value in the form of working product increments or features. By breaking down the project into manageable iterations (sprints), teams can continuously deliver tangible results, enabling stakeholders to provide feedback and make timely adjustments as needed.
- **Collaboration and Transparency:** Agile fosters a collaborative and transparent work environment. Through regular meetings such as daily stand-ups, sprint planning, and sprint reviews, team members actively communicate and collaborate, sharing progress, challenges, and insights. This promotes a shared understanding, effective problem-solving, and collective ownership of the project's success.
- **Early and Regular Stakeholder Involvement:** Agile encourages frequent stakeholder involvement throughout the project. By engaging stakeholders in the review and feedback process during sprint reviews, the project benefits from their input, ensuring that the delivered product meets their expectations and requirements.
- **Continuous Improvement:** Agile embraces the principle of continuous improvement. Through retrospectives held at the end of each sprint, teams reflect on their work, identify areas for improvement, and implement changes in subsequent sprints. This iterative approach fosters a culture of learning and adaptation, driving the project towards greater efficiency and effectiveness.

1.2.2 KDD process

This comprehensive approach comprises several stages, each playing a crucial role in unearthing valuable insights from a database.
The KDD[6] process is structured around five distinct stages, as depicted in the figure below.

1. **Selection:** In the first stage, a target data set or a subset of variables/samples is chosen for knowledge discovery. This selection stage determines the data subset for further analysis.
2. **Preprocessing:** The preprocessing stage involves cleaning and preparing the selected target data set to ensure data consistency and quality.

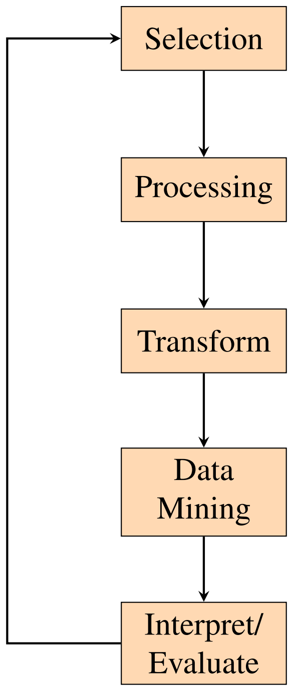


Figure 1.3: KDD Process Steps

3. **Transformation:** During the transformation stage, the data undergoes dimensionality reduction or transformation methods.

4. **Data Mining:** The data mining stage entails the search for patterns of interest in the transformed data. Depending on the objective of data mining, which typically revolves around prediction, various algorithms and techniques are applied to uncover meaningful patterns, associations, correlations, or predictive models.

5. **Interpretation/Evaluation:** The final stage of the KDD process involves the interpretation and evaluation of the mined patterns. The discovered patterns are analyzed, interpreted, and assessed in terms of their relevance, usefulness, and potential impact.

Through the systematic execution of these stages, the KDD process enables the extraction of knowledge from databases, providing valuable insights and actionable information. It serves as a structured framework for effectively uncovering patterns and generating meaningful knowledge that can be utilized for decision-making, prediction, and other data-driven tasks.

1.2.3 CRISP-DM Process

The CRISP-DM[16] (Cross-Industry Standard Process for Data Mining) process provides a structured and comprehensive framework for organizations to conduct data mining projects effectively and efficiently. The CRISP-DM methodology consists of six iterative stages that cover the entire data mining lifecycle, from understanding the business problem to deploying the results. These stages are as follows:

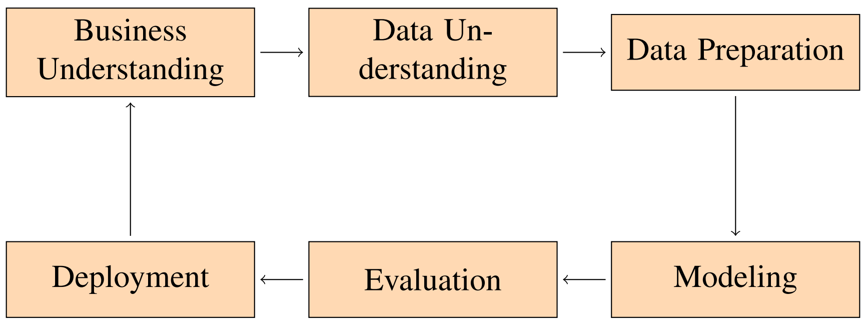


Figure 1.4: CRISP-DM Process Steps

1. **Business Understanding:** This initial stage focuses on understanding the business objectives, goals, and requirements of the data mining project.

2. **Data Understanding:** In this stage, data collection and exploration take place. The project team assesses the available data sources, collects the relevant data, and performs an initial analysis to gain a better understanding of the data's structure, quality, and potential challenges.

3. **Data Preparation:** The data preparation stage involves cleaning, integrating, and transforming the collected data to create a dataset suitable for data mining.

4. **Modeling:** During the modeling stage, various data mining techniques and algorithms are applied to the prepared dataset. The goal is to build predictive models, uncover patterns, or discover relationships within the data. Multiple models may be created and evaluated to find the most effective one for the given problem.

5. **Evaluation:** In the evaluation stage, the performance and effectiveness of the developed models are assessed.

6. **Deployment:** The final stage involves deploying the selected model or solution into the operational environment.

CRISP-DM's iterative nature allows for feedback and iteration between stages, enabling refinement and improvement throughout the data mining process.

1.2.4 SEMMA Process

The SEMMA methodology, developed by SAS Institute, stands for Sample, Explore, Modify, Model, and Assess. It offers a systematic approach to data mining, guiding analysts through each project stage. Here's an overview of each step.

1. **Sample:**

   In the sampling stage, a representative subset of the available data is selected for analysis.

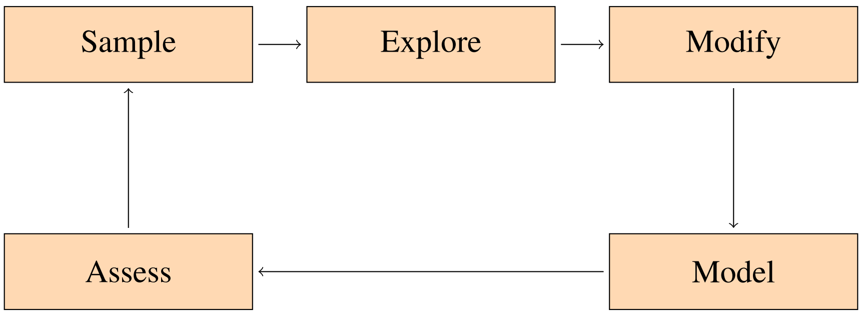


Figure 1.5: SEMMA Process Steps

2. **Explore:**

   The exploration stage involves understanding the data through descriptive statistical techniques and data visualization.

3. **Modify:**

   In the modification stage, data is prepared and transformed to improve its quality and relevance for modeling.

4. **Model:**

   The modeling stage focuses on the development and application of predictive models or other analytical techniques to extract insights from the data.

5. **Assess:**

   The assessment stage involves evaluating the models' performance and assessing their usefulness for the intended purpose. This includes evaluating predictive accuracy, assessing model interpretability, and considering the models' business or practical value.

1.2.5 Comparative Study

Comparing the stages of KDD (Knowledge Discovery in Databases) and SEMMA (Sample, Explore, Modify, Model, Assess)[2], it appears that they align in a corresponding manner. For instance, the Sample stage in SEMMA can be equated to the Selection stage in KDD, while Explore in SEMMA can be linked to Preprocessing, Modify to Transformation, Model to Data Mining, and Assess to Interpretation/Evaluation. However, a more comprehensive examination reveals that SEMMA's five stages can be viewed as a practical implementation of KDD's stages, particularly when considering their direct association with the SAS Enterprise Miner software.

In contrast, comparing the stages of KDD with those of CRISP-DM (Cross-Industry Standard Process for Data Mining) is not as straightforward as in the SEMMA scenario. Nonetheless, it is noteworthy that the CRISP-DM methodology encompasses steps that must precede and follow the KDD process. For instance, the Business Understanding phase in CRISP-DM aligns with developing an understanding of the application domain, relevant prior knowledge, and the end-users goals. Similarly, the Deployment phase in CRISP-DM can be identified as the consolidation of this knowledge into the system, ensuring its effective integration and utilization.

Step	Methodology		
	CRISP-DM	KDD	SEMMA
Business Understanding	Yes	No	No
Data Understanding	Yes	Yes	Yes
Data Preparation	Yes	Yes	Yes
Model Building	Yes	Yes	Yes
Model Evaluation	Yes	Yes	Yes
Model Deployment	Yes	No	No
Sample	No	Yes	Yes
Explore	No	Yes	Yes
Modify	No	Yes	Yes

Table 1.1: Comparative Table of Data Mining Methodologies and Steps

1.2.6 Decision

In our project, the strengths of CRISP-DM shine through, particularly in the areas of business understanding and deployment. By following the CRISP-DM methodology, we ensure a comprehensive understanding of the project's business context, goals, and requirements. This enables us to accurately define the data mining problem and align our efforts with the needs of the business. Additionally, the deployment stage ensures that the knowledge and insights gained from the project are effectively organized, presented, and utilized by the intended users or stakeholders.

1.3 Project Implementation and Planning

Here, we present two essential elements that guide our project's progress

1.3.1 Gantt Chart

Through these visual representations, we provide a comprehensive overview of our project's timeline.

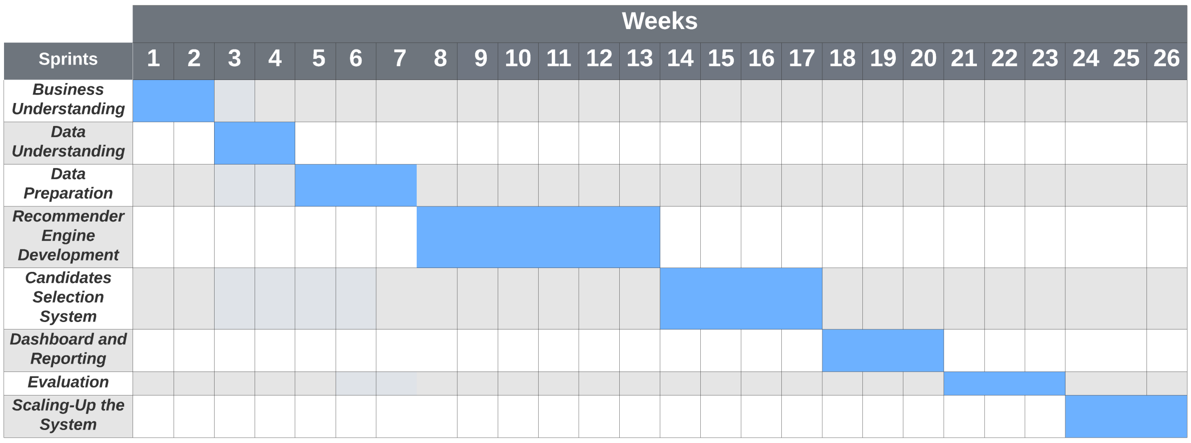


Figure 1.6: Gantt Chart

1.3.2 Communication and Collaboration

In our project, effective communication and collaboration are key components that contribute to the overall success and efficiency of the development process. Throughout each sprint, we prioritize validation and communication within the team to ensure that we are on track and aligned with project objectives. Microsoft Teams serves as our primary communication platform, providing a centralized space for sharing updates, discussing challenges, and collaborating on solutions.

During our sprint validations, we come together as a team to review the progress made, evaluate the completed tasks, and ensure that we are meeting the sprint goals. This collaborative validation process allows us to identify any deviations, address potential issues promptly, and make necessary adjustments to maintain project momentum. By validating our work regularly, we ensure that we are consistently delivering value and staying on track towards achieving the project milestones.

Microsoft Teams plays a vital role in facilitating effective communication among team members. We utilize dedicated channels within Teams to share our challenges, ideas, and solutions. This platform allows us to engage in real-time discussions, enabling everyone to contribute their expertise and insights. By leveraging this shared space, we foster an environment of transparency, active participation, and collective problem-solving. Each team member has the opportunity to contribute their perspectives and propose solutions, leading to more efficient progress in the project.

1.4 Technologies and Tools Employed in this Project

In the development of our AI system for this project, Python emerged as the language of choice. Python offers a rich ecosystem of machine learning and natural language processing libraries, making it well-suited for AI-related tasks. The availability of popular libraries such as TensorFlow, Scikit-learn, and NLTK (Natural Language Toolkit) provided us with a powerful framework for implementing advanced algorithms and models. One significant advantage of Python is its interpreted nature. This means that we can execute the code without the need for compilation, allowing for quick iterations and code modifications. The ability to make changes on the fly accelerates the development process, enabling us to iterate rapidly and experiment with different approaches and techniques.

Python also excels in its capability to seamlessly integrate with other languages and platforms. Through the use of RESTful APIs, we can establish connections and facilitate data exchange with external systems or programming languages. This flexibility enables us to leverage the strengths of other tools, languages, or services, expanding the functionality and interoperability of our AI system.

For the crucial tasks of data integration and cleaning, we employed Talend Open Studio. Talend is a robust and versatile data integration tool that offers comprehensive features for data extraction, transformation, and loading (ETL). With Talend, we were able to streamline the process of integrating data from various sources, performing data transformations, and ensuring data quality and consistency.

To present our findings and insights in a visually appealing and intuitive manner, we utilized Power BI for reporting purposes. Power BI is a powerful business intelligence tool that enables us to create interactive dashboards, reports, and visualizations. With its user-friendly interface and extensive visualization options, we were able to convey complex information and analysis results in a clear and concise manner, facilitating effective decision-making and communication with stakeholders.

In addition to the technologies and tools mentioned earlier, MySQL was selected as the database management system for this project. MySQL is a widely-used open-source relational database management system known for its reliability, scalability, and performance.

The choice of MySQL as the database solution was driven by several factors. Firstly, MySQL offers excellent compatibility with Python, making it seamless to establish connections and perform database operations within our Python-based AI system. This compatibility facilitated the integration of data retrieval, storage, and manipulation functionalities seamlessly. Moreover, MySQL has strong community support.

Conclusion

In conclusion, the first chapter of this thesis provided an overview of the project, its objectives, and the problems it aims to address which are all what we need to establish a strong Business Understanding of the project. We discussed the significance of developing a data-driven solution leveraging advanced recommender systems and AI algorithms to provide personalized job recommendations and assist recruiters in finding the best candidates.

Throughout the project, we will follow an Agile Scrum methodology, conducting the work in sprints. Each chapter of the thesis will correspond to a sprint, allowing for a detailed examination of the activities, progress, and outcomes achieved during that specific phase.

The chapter covered in this sprint highlighted the benchmark of the technologies and tools used, including Python for AI system development due to its rich ML and NLP libraries, its interpretive nature, and seamless integration capabilities. Talend Open Studio was employed for data integration and cleaning, while Power BI served as the reporting tool for visualizing and presenting findings.

The subsequent chapters of this thesis will delve into each sprint, exploring its steps, analysis, and implementation.

Chapter 2

Data Understanding

Introduction

The ”Data Understanding” chapter focuses on gaining a comprehensive understanding of the data involved in the project. This chapter plays a crucial role in laying the groundwork for subsequent data analysis and modeling tasks.

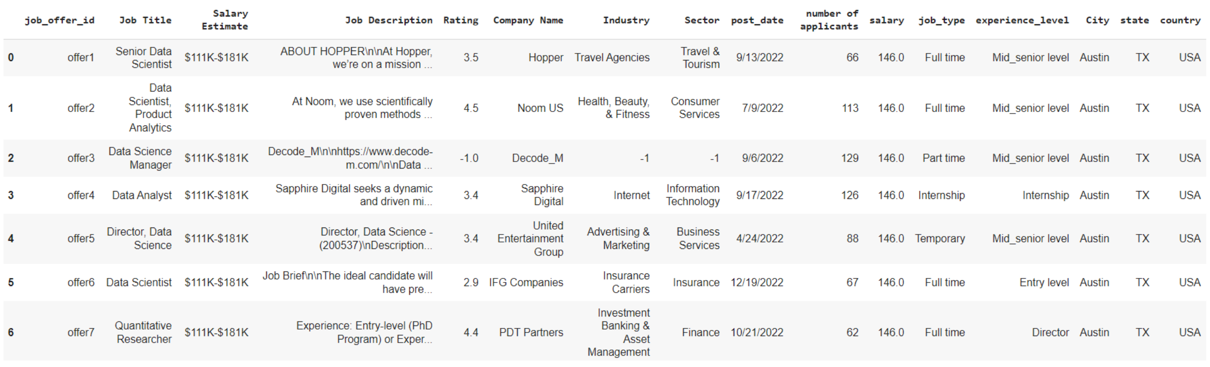
2.1 Background and Context

It is important to note that the data utilized in this project is publicly available and obtained from reliable sources on the Internet.
The decision to use public data stems from the need to work with real-world information that is accessible to the wider research community. By utilizing publicly available data, we ensure that the findings and insights derived from our analysis can be validated and reproduced by others in the field.
To gain a deeper understanding of the data, we will conduct an Exploratory Data Analysis (EDA). Through EDA, we aim to uncover valuable insights, detect outliers, understand the distribution of variables, and identify any missing or inconsistent values.
The data used in this project can be categorized into two main data sets: job offer data and job seeker data.
The job offer data provides insights into the various job opportunities available in the market. It includes details such as the location of the job, job titles, descriptions, salary ranges, required experience, and other relevant attributes. This data is sourced from reputable job portals, industry-specific websites such as Glassdoor, LinkedIn, and Indeed.
On the other hand, job seeker data focuses on individuals seeking employment. This data encompasses information related to job seekers, including their resumes, job titles, fields of expertise, age, salary expectations, and other pertinent details. The job seeker data is typically sourced through surveys or online profiles. Analyzing this data provides insights into the characteristics, preferences, and qualifications of job seekers, allowing us to match their profiles

with suitable job opportunities.
By dividing the data into these two major data sets, we can effectively analyze and address the needs of both job seekers and employers. Understanding the job offer data enables us to provide personalized job recommendations to job seekers based on their qualifications, preferences, and the specific requirements of available job opportunities. Simultaneously, the job seeker data helps us assess the profiles and qualifications of candidates and assists recruiters in identifying the most suitable candidates for their job openings.

2.1.1 Job Offers Data

The job seekers' data utilized in this project is structured and presented in a tabular format, specifically in the .csv (Comma-Separated Values) file format. The use of a tabular structure allows for an organized and systematic representation of the data, making it easier to analyze, manipulate, and extract meaningful insights.
The screenshot below showcases our data in the format of a Pandas Dataframe:
Here's a brief description of each column:



	job_offer_id	Job Title	Salary Estimate	Job Description	Rating	Company Name	Industry	Sector	post_date	number of applicants	salary	job_type	experience_level	City	state	country
0	offer1	Senior Data Scientist	$111K-$181K	ABOUT HOPPER\n\nAt Hopper, we're on a mission ...	3.5	Hopper	Travel Agencies	Travel & Tourism	9/13/2022	66	146.0	Full time	Mid_senior level	Austin	TX	USA
1	offer2	Data Scientist, Product Analytics	$111K-$181K	At Noom, we use scientifically proven methods ...	4.5	Noom US	Health, Beauty, & Fitness	Consumer Services	7/9/2022	113	146.0	Full time	Mid_senior level	Austin	TX	USA
2	offer3	Data Science Manager	$111K-$181K	Decode_M\n\nhttps://www.decode-m.com/\n\nData ...	-1.0	Decode_M	-1	-1	9/6/2022	129	146.0	Part time	Mid_senior level	Austin	TX	USA
3	offer4	Data Analyst	$111K-$181K	Sapphire Digital seeks a dynamic and driven mi...	3.4	Sapphire Digital	Internet	Information Technology	9/17/2022	126	146.0	Internship	Internship	Austin	TX	USA
4	offer5	Director, Data Science	$111K-$181K	Director, Data Science - (200537)\nDescription...	3.4	United Entertainment Group	Advertising & Marketing	Business Services	4/24/2022	88	146.0	Temporary	Mid_senior level	Austin	TX	USA
5	offer6	Data Scientist	$111K-$181K	Job Brief\n\nThe ideal candidate will have pre...	2.9	IFG Companies	Insurance Carriers	Insurance	12/19/2022	67	146.0	Full time	Entry level	Austin	TX	USA
6	offer7	Quantitative Researcher	$111K-$181K	Experience: Entry-level (PhD Program) or Exper...	4.4	PDT Partners	Investment Banking & Asset Management	Finance	10/21/2022	62	146.0	Full time	Director	Austin	TX	USA

Figure 2.1: Job Offers Dataframe

1. **"job_offer_id" (string)**
   - This column represents a unique identifier for each job offer in the dataset. It serves as a reference point to distinguish one job offer from another.
2. **"Job Title" (string)**
   - This column contains the job title or position associated with each job offer. It provides information about the specific role or position to which the offer pertains to.
3. **"Salary Estimate" (string)**
   - This column presents the salary range estimate for the job offer. It indicates the expected salary range for the position, providing insights into the potential compensation.

4. **”Job Description” (raw text)**
    - This column provides the raw text of the job description associated with the job offer. It includes detailed information about the responsibilities, requirements, and qualifications for the position.
5. **”Rating” (float)**
    - This column represents the rating associated with the job offer. It indicates the rating, or score assigned to the job based on various factors, potentially reflecting the reputation or desirability of the company or position.
6. **”Company Name” (string)**
    - This column contains the name of the company offering the job. It identifies the organization associated with the job offer.
7. **”Industry” (string)**
    - This column specifies the industry to which the job offer belongs. It categorizes the job offer based on the sector or field of business.
8. **”Sector” (string)**
    - This column provides the sector or domain associated with the job offer. It further categorizes the job offer within the broader industry classification.
9. **”post_date” (date)**
    - This column represents the date on which the job offer was posted. It indicates when the offer became available for applicants to view and apply.
10. **”number of applicants” (integer)**
    - This column denotes the number of applicants who have applied for the specific job offer. It provides insights into the level of competition or interest generated by the position.
11. **”salary” (float)**
    - This column represents the calculated average salary for the job offer. It is derived by averaging the lower and upper bounds of the salary range provided in the ”Salary Estimate” column.
12. **”Job Type” (string)**
    - This column specifies the type or category of the job offer, such as full-time, part-time, contract, or internship. It describes the employment arrangement associated with the position.

13. **”experience_level” (categorical)**

   - This column represents the experience level required for the job offer. It categorizes the level of experience expected from applicants.

14. **”city” (string)**

   - This column specifies the city associated with the location of the job offer. It indicates the specific urban area where the position is based.

15. **”state” (string)**

   - This column represents the state or region associated with the location of the job offer. It provides information about the broader geographic area of the position.

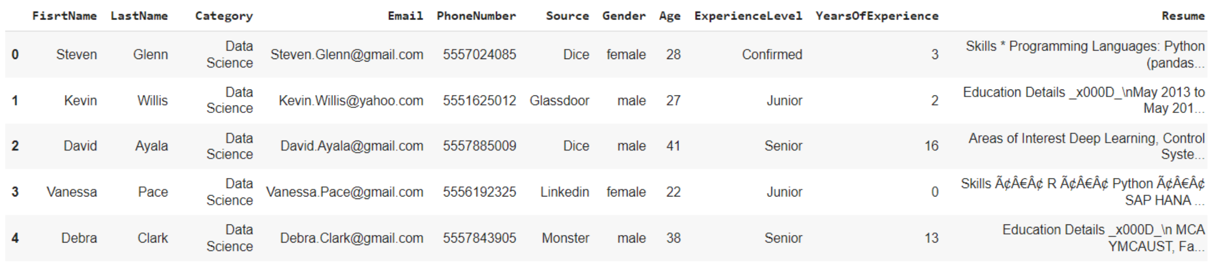
16. **”country” (string)**

   - This column indicates the country associated with the location of the job offer. It specifies the nation in which the position is available.

The Job Offers dataset comprises a vast collection of more than 60,000 IT job offers and encompasses 16 features that provide valuable information and insights regarding these job opportunities.

2.1.2 Job Seekers Data

The job offers data used on this project is also structured and presented in a tabular format, specifically in the .csv (Comma-Separated Values) file format.
The screenshot below showcases our data in the format of a Pandas Dataframe:



	FisrtName	LastName	Category	Email	PhoneNumber	Source	Gender	Age	ExperienceLevel	YearsOfExperience	Resume
0	Steven	Glenn	Data Science	Steven.Glenn@gmail.com	5557024085	Dice	female	28	Confirmed	3	Skills * Programming Languages: Python (pandas...
1	Kevin	Willis	Data Science	Kevin.Willis@yahoo.com	5551625012	Glassdoor	male	27	Junior	2	Education Details _x000D_\nMay 2013 to May 201...
2	David	Ayala	Data Science	David.Ayala@gmail.com	5557885009	Dice	male	41	Senior	16	Areas of Interest Deep Learning, Control Syste...
3	Vanessa	Pace	Data Science	Vanessa.Pace@gmail.com	5556192325	Linkedin	female	22	Junior	0	Skills Ã¢Â€Â¢ R Ã¢Â€Â¢ Python Ã¢Â€Â¢ SAP HANA ...
4	Debra	Clark	Data Science	Debra.Clark@gmail.com	5557843905	Monster	male	38	Senior	13	Education Details _x000D_\n MCA YMCAUST, Fa...

Figure 2.2: Job Seekers Dataframe

Here’s a brief description of each column:

1. **”FirstName” (String)**
   - This column contains dummy data representing the first name of the job seeker. It is used for identification purposes and does not correspond to real individuals.
2. **”LastName” (String)**
   - This column contains dummy data representing the last name of the job seeker. Similar to ’FirstName’, it is used for identification purposes and does not correspond to real individuals.
3. **”Category” (String)**
   - This column specifies the category or job title of the job seeker. It represents the specific role or position the individual is seeking, such as data scientist, web developer, or any other relevant job title.
4. **”Email” (String)**
   - This column contains dummy data representing the job seeker’s email address. It is used for identification purposes and does not correspond to real email addresses.
5. **”PhoneNumber” (String)**
   - This column contains dummy data representing the phone number of the job seeker. Similar to ’Email’, it is used for identification purposes and does not correspond to real phone numbers.
6. **”Source” (String)**
   - This column indicates the source through which the job seeker came across the job opportunity. It could include platforms such as LinkedIn, Indeed, company websites, or any other relevant sources.
7. **”Gender” (String, Categorical)**
   - This column captures the gender of the job seeker. It indicates the individual’s gender identity, such as male, female, or other gender identities.
8. **”Age” (int)**
   - This column represents the age of the job seeker. It specifies the individual’s age at the time of their application or inclusion in the dataset.
9. **”ExperienceLevel” (String, Categorical)**

- This column denotes the experience level of the job seeker. It categorizes the individual's level of professional experience, such as entry-level, mid-level, senior, or any other relevant experience categories.

10. **"YearsOfExperience" (int)**
    - This column quantifies the number of years of professional experience the job seeker possesses. It represents the length of time the individual has been working in their field or industry.

11. **"Resume" (String)**
    - This column contains the job seeker's resume or curriculum vitae (CV). It includes detailed information about the individual's educational background, work experience, skills, certifications, and other relevant qualifications.

Interpretation

By examining and understanding the job seekers and job offers data, we comprehensively understood the attributes and information available for further analysis and decision-making.

2.2 Implementation and Execution

Here, we outline data exploration and analysis methods, including tools and languages for enhancing data understanding.
So, in the following, we highlight the steps taken to gain insights into the data.

Data Loading:
- To initiate the data understanding process, the dataset was loaded into Python using the powerful Pandas library. This facilitated efficient data manipulation and analysis by converting the dataset into a Pandas Dataframe, a versatile data structure widely used for working with structured data.

Dataframe Shape:
- An essential aspect of data understanding is assessing the dimensions of the Dataframe. By checking the shape of the Dataframe, we obtained valuable information about the dataset's size, including the number of rows and columns. Understanding the dataset's dimensions set the foundation for subsequent exploration and analysis.

Missing Values:
- To ensure the quality of the dataset, we meticulously examined the Dataframe for any missing values using Pandas. Identifying and addressing missing data is crucial, as it impacts the accuracy and reliability of subsequent analysis and modeling tasks. Through this process, we gained a comprehensive view of any missing values present in the dataset.

Data Types: • An in-depth understanding of the data types within the Dataframe is paramount for guiding appropriate data manipulation and analysis techniques. We thoroughly examined the data types of each attribute in the Dataframe, such as numeric, categorical, or text data. This enabled us to tailor our analysis approach to suit the characteristics of each attribute.

Data Description: • Methods: To gain deeper insights into the dataset, various data description methods were employed. Summary statistics, including measures such as mean, median, and standard deviation, were calculated for numerical attributes. These statistics provided a comprehensive overview of the central tendencies and variability of the data, aiding in understanding its distribution.

Data Visualization: • Data visualization plays a pivotal role in uncovering patterns, trends, and relationships within the dataset. We leveraged powerful data visualization libraries, such as Seaborn and WordCloud, to create meaningful visual representations. Histograms, box plots, and word clouds were among the visualizations utilized to gain valuable insights into the dataset, enhancing our understanding of its characteristics.

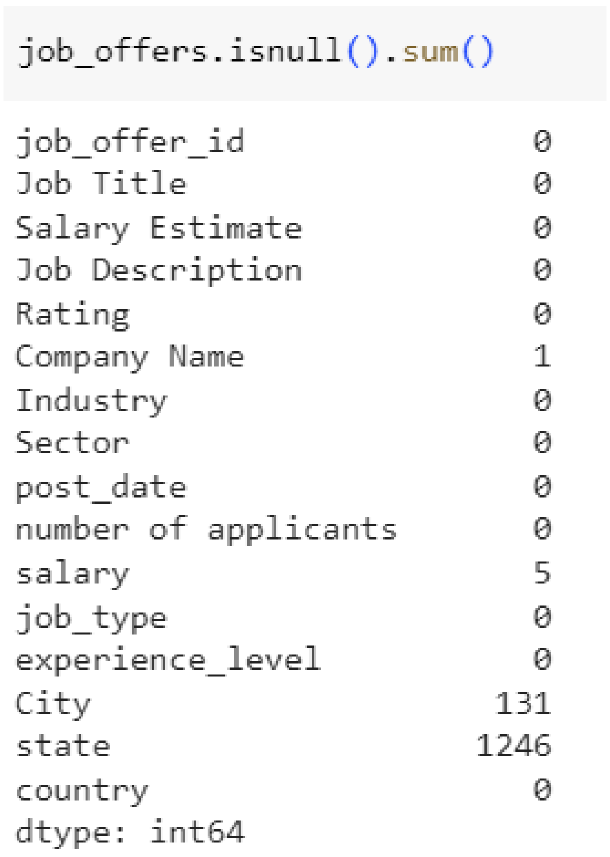
Programming Language and Software Used: • Python, renowned for its versatility and extensive data analysis capabilities, served as the primary programming language throughout the data understanding phase. Complemented by tools such as Jupiter Notebook and the Anaconda distribution, Python empowered us to seamlessly execute data manipulation, statistical analysis, and visualization tasks.

2.3 Results and Finding

Here, we present the results and findings obtained from the data understanding phase. We discuss the data profiling activities, descriptive statistics, and visualizations created to gain insights into the data.

2.3.1 Job Offers Data Exploration

This visualization below shows a clear overview of the columns that contain missing values and allows us to assess the extent of missing data in each attribute.



```
job_offers.isnull().sum()
```

```
job_offer_id                 0
Job Title                    0
Salary Estimate              0
Job Description              0
Rating                       0
Company Name                 1
Industry                     0
Sector                       0
post_date                    0
number of applicants         0
salary                       5
job_type                     0
experience_level             0
City                       131
state                     1246
country                      0
dtype: int64
```

Figure 2.3: Job Offers Null Values

Our findings revealed the presence of null values in specific columns. Here is a summary of the null values identified:

- **’Company Name’:** One null value was observed in the ’Company Name’ column. This indicates that there is one job offer in the dataset where the company name is not provided.
- **’City’:** The ’City’ attribute contained 131 null values. These null values indicate that there are 131 job offers where the city information is missing or not provided.
- **’State’:** The ’State’ attribute exhibited 1246 null values. These null values indicate that there are 1246 job offers where the state information is missing or not provided.

By identifying the columns with null values, we can take appropriate steps to handle them during the data preprocessing phase. It enables us to ensure the integrity and reliability of the data for subsequent analysis and modeling tasks.
Now, we present a pie chart showcasing the distribution of the top 10 job titles in the job offers dataset. This chart provides insights into the demand for different job titles in the field and reflects real-life trends.
The Pie Chart illustrating the distribution of the top 10 job titles in the job offers dataset is displayed below:

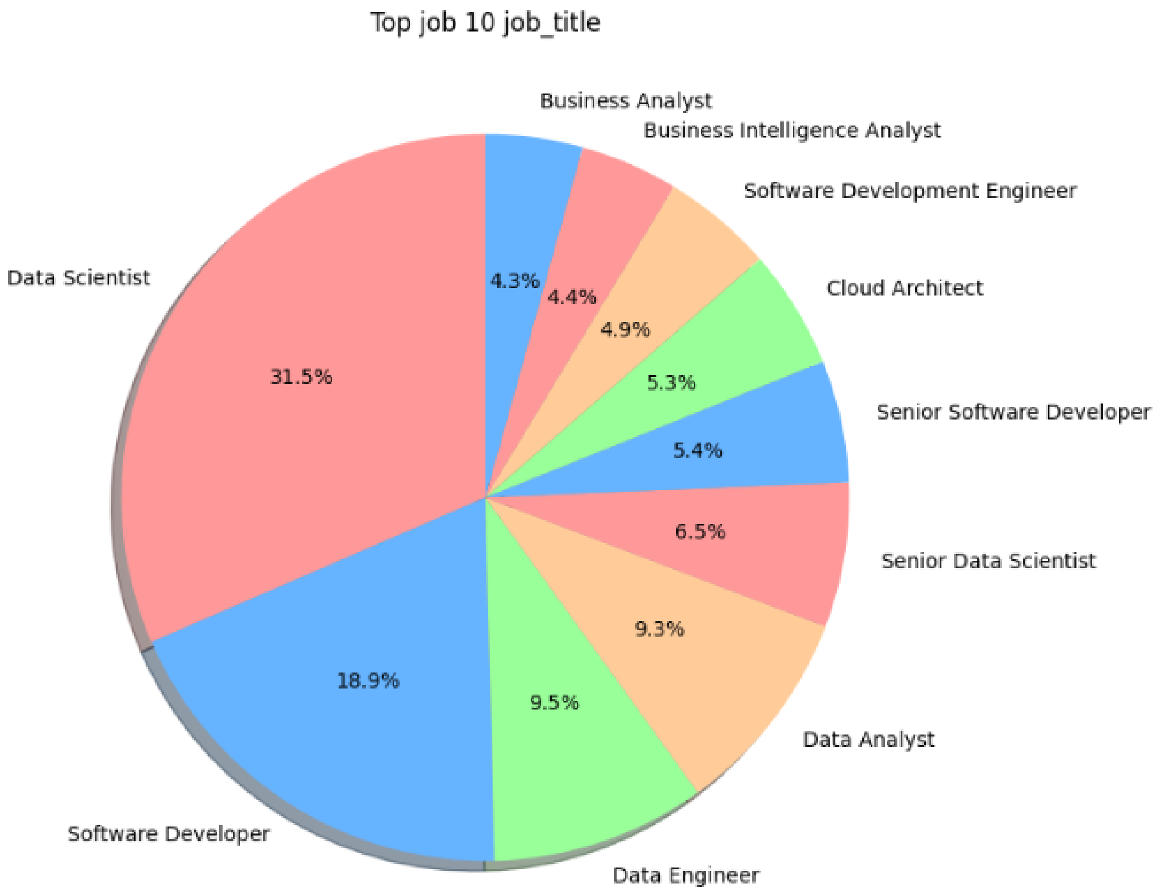


Figure 2.4: Top 10 Job Titles

The pie chart highlights the relative proportions of the top 10 job titles based on their occurrence in the dataset. The data reveals that the job title 'Data Scientist' occupies the largest portion, indicating its high demand in the job market. Following closely, 'Software Developer' holds the second-largest slice, signifying its prominence in the industry. In the third position, we find 'Data Engineer', reflecting the growing demand for professionals with expertise in managing and processing large volumes of data.
The remaining job titles in the top 10 encompass a diverse range of roles, including 'Business Analyst', Data Analyst, 'Cloud Architect', and 'Business Analyst'. These job titles highlight the varied skill sets and specialized roles sought by employers in the industry.

To understand the distribution of the number of applicants for each job offer, we created a visualization that showcases the distribution pattern. The screenshot below displays the distribution plot:

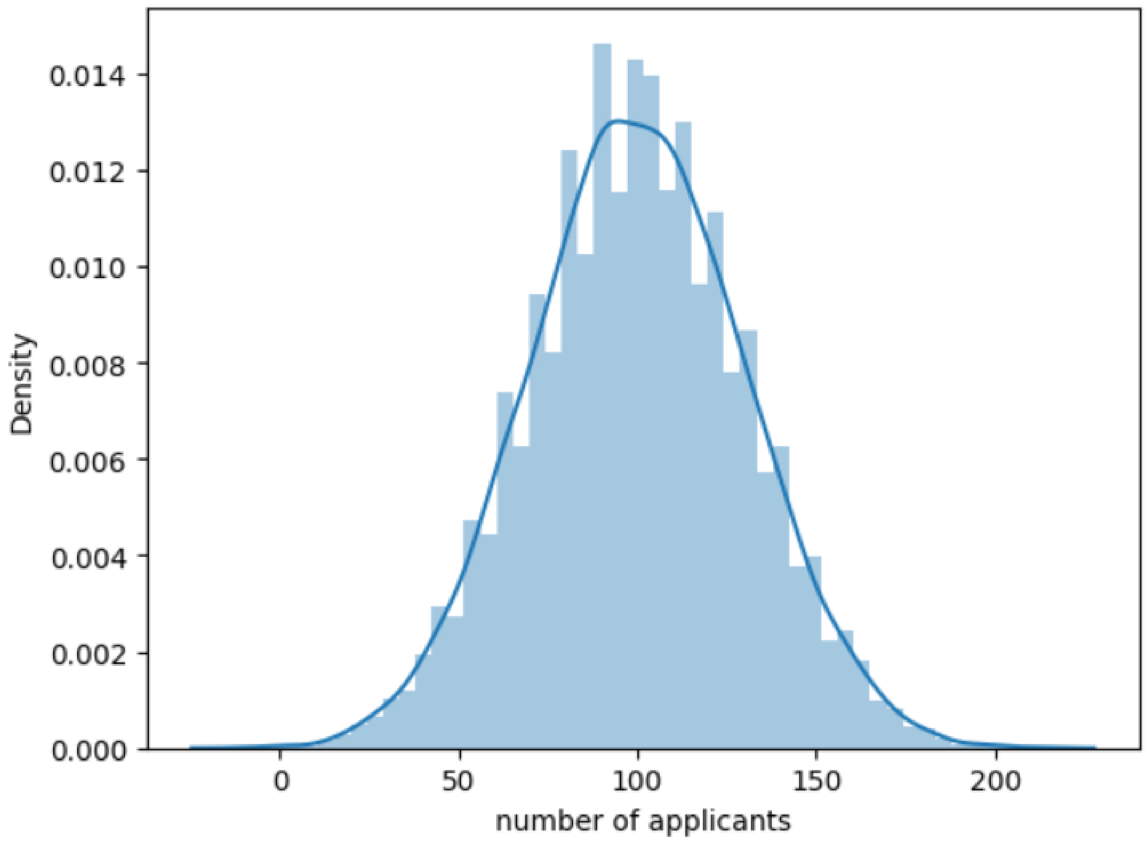
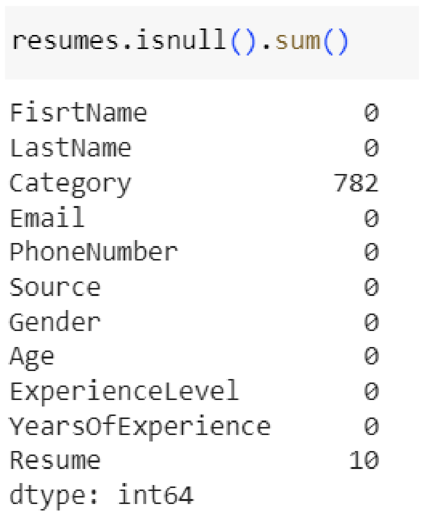


Figure 2.5: Applications Distribution

The distribution plot illustrates the number of applicants for each job offer, with the x-axis representing the number of applicants and the y-axis indicating the frequency of job offers. In this particular case, the distribution follows a normal distribution pattern, with a mean value of 100 applicants.
The normal distribution shape provides valuable insights into the competitiveness of the job offers and the level of interest they generated among potential applicants. It helps us understand the typical range and spread of the number of applicants, allowing us to gauge the popularity and demand for different job opportunities.
By analyzing this distribution, we can identify job offers that received a higher or lower number of applicants relative to the mean value. This information can be valuable for employers and job seekers alike, offering insights into the level of competition and the potential attractiveness of different job offers.

2.3.2 Job Seekers Data Exploration

To ensure data quality and comprehensiveness, we examined the job seekers' dataset for any missing values. The screenshot below illustrates the presence of null values in each column:



```
resumes.isnull().sum()
```

```
FisrtName            0
LastName             0
Category           782
Email                0
PhoneNumber          0
Source               0
Gender               0
Age                  0
ExperienceLevel      0
YearsOfExperience    0
Resume              10
dtype: int64
```

Figure 2.6: Job Seekers Null Values

Upon analysis, we found that the 'Category' column contains 782 null values. These null values indicate that 782 job seekers' entries do not have a specified job category or title.
Furthermore, the 'Resume' column exhibits 10 null values, suggesting that 10 job seekers' entries lack a submitted resume.
To gain insights into the distribution of job seekers' titles or categories, we created a pie chart visualization.
The screenshot below showcases the pie chart, highlighting the top 10 job seekers' titles or categories:

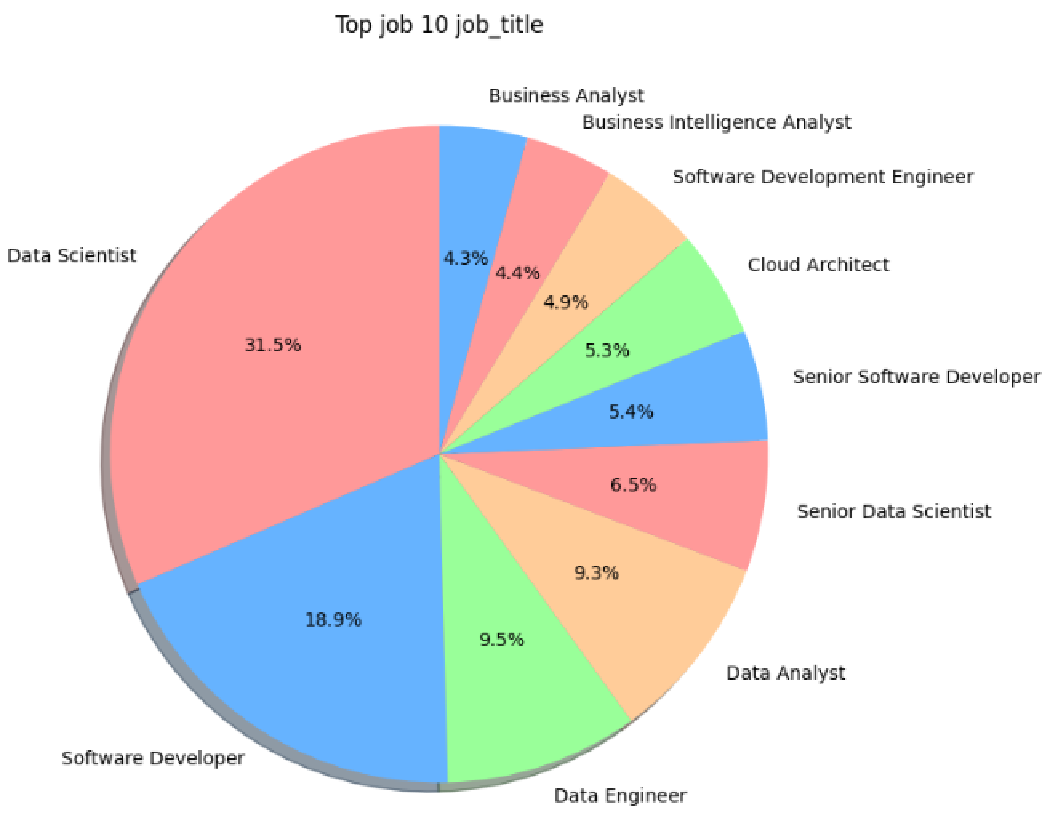


Figure 2.7: Top 10 Job Seekers Profiles

The pie chart illustrates the relative proportions of the top 10 job seekers' titles or categories based on their occurrence in the dataset. Among the top titles, 'Software Developer' holds the highest share at 19.5%, reflecting its prominence in the job seekers' dataset. Following closely, 'Systems Administrator' accounts for 14.0% of the titles, indicating its significant representation among job seekers. In the third position, 'Project Manager' holds 11.8%, further emphasizing its prevalence in the dataset.

To understand the distribution of resume lengths among job seekers, we created a line chart visualization.
The screenshot below displays the line chart, illustrating the frequency distribution of resume lengths:

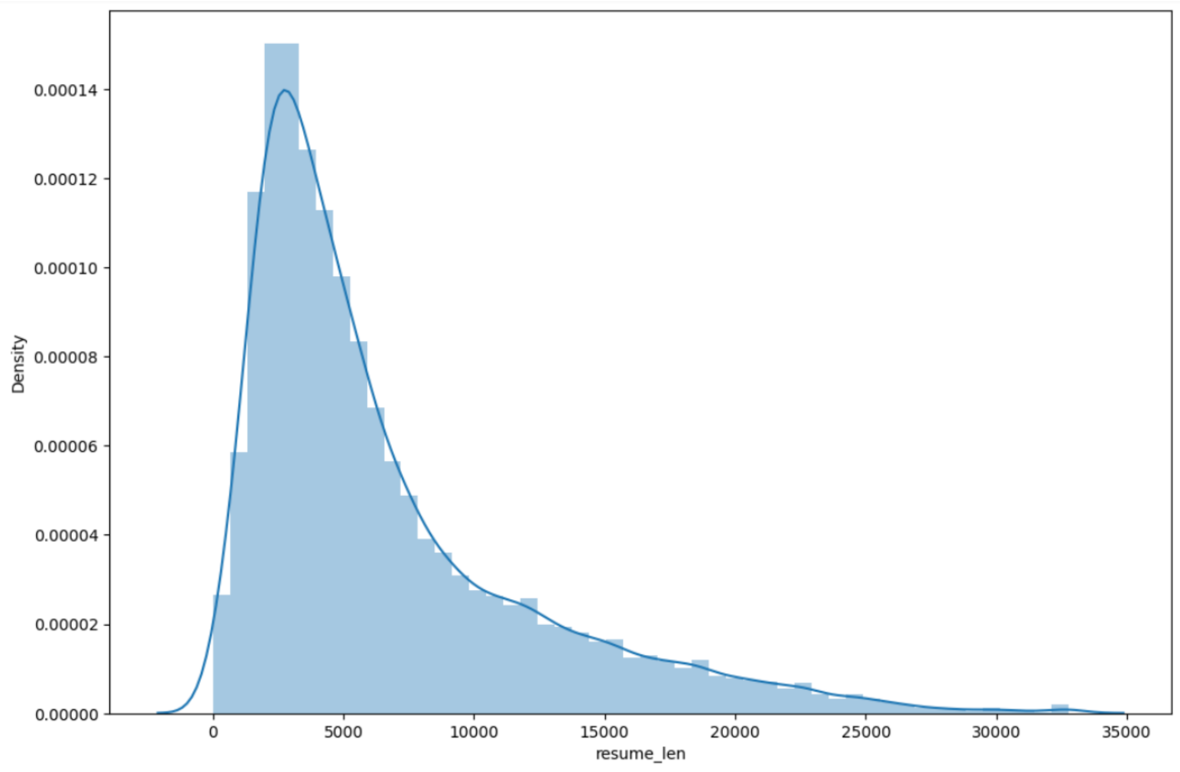


Figure 2.8: Job Seekers Resume Lenght Distribution

The x-axis represents the length of resumes, while the y-axis indicates the frequency of resumes with a specific length. The line chart exhibits a characteristic bell-shaped curve, indicative of a normal distribution pattern. The peak of the curve represents the most common resume length, while the tails on either side indicate rarer lengths.
Analyzing the line chart allows us to identify the average or typical resume length, which aligns with the peak of the curve. This insight can be valuable for employers and recruiters, providing an understanding of the common length of resumes they may encounter during the hiring process.
Now, to examine the age distribution of job seekers, we created a line chart visualization. The screenshot below displays the line chart, presenting the frequency distribution of ages:

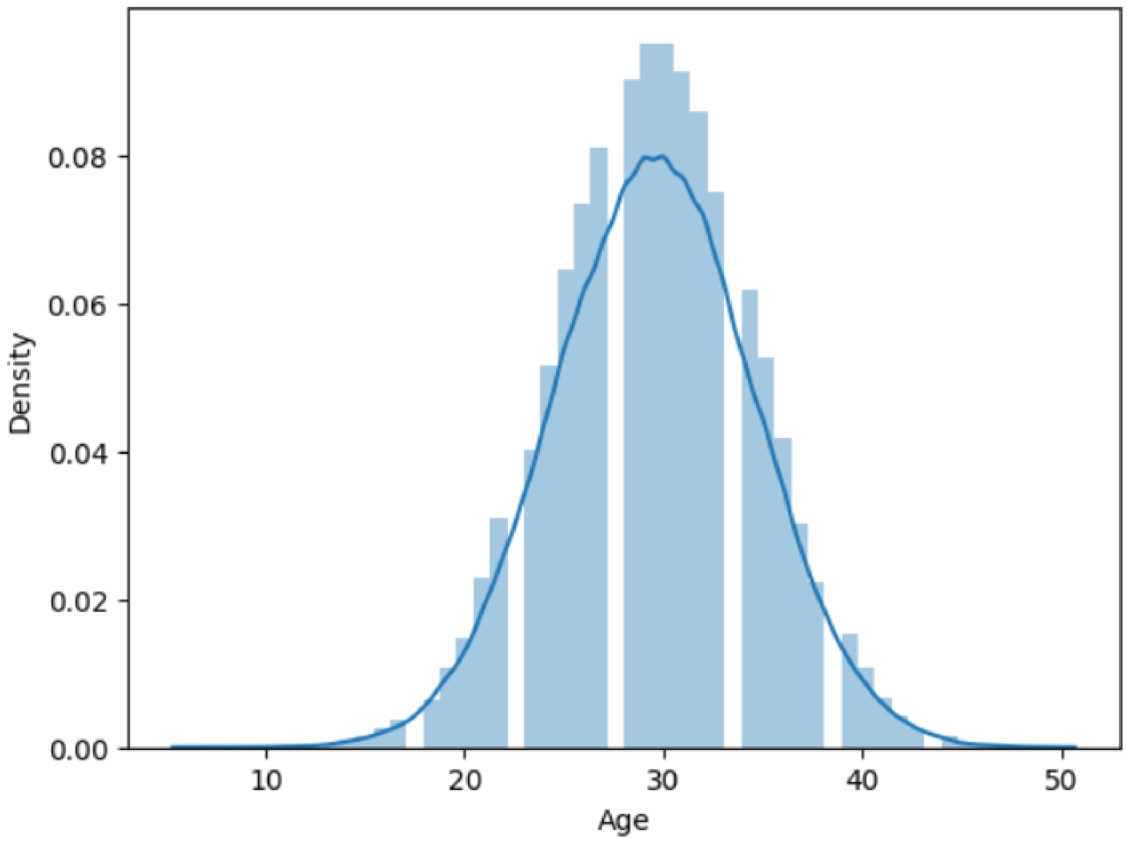


Figure 2.9: Job Seekers Age Distribution

The x-axis represents the age of job seekers, while the y-axis indicates the frequency of job seekers at specific age intervals. The line chart exhibits a bell-shaped curve, characteristic of a normal distribution pattern. The peak of the curve indicates the most common age among job seekers, which aligns with the mean age of 30.
Analyzing the line chart allows us to gain insights into the typical age profile of job seekers in the dataset. It helps us understand the age distribution and identify the age group that is most prevalent among potential candidates.

2.4 Lesson Learned and Reflection

In the course of exploring the job seekers' and job offers' datasets, several valuable lessons have been learned, and reflections have emerged. These insights have contributed to a deeper understanding of the data and its implications for the project's objectives. Here are the key lessons learned and reflections from the data exploration:

- **Data Completeness and Quality are Vital:** During the exploration of both datasets, it became evident that data completeness and quality significantly impact the effectiveness of subsequent analysis and modeling. Identifying and handling missing values in the datasets is crucial to ensure accurate and reliable insights.

- **Understanding Real-Life Trends:** The analysis of the top job titles or categories among job seekers and the demand for different job titles in the job offers dataset provided valuable insights into real-life trends. These trends mirror the current market demands and can assist in aligning the project's objectives with prevailing job opportunities.
- **Leveraging Data Visualization:** Utilizing various data visualization techniques, such as pie charts and line charts, allowed us to effectively communicate complex information and patterns in an intuitive manner. These visualizations offer a clear and concise representation of the data, making it easier to identify trends and draw meaningful conclusions.
- **Normal Distribution Observations:** The observation of normal distribution patterns in the age and resume length distribution highlights the significance of understanding the underlying statistical characteristics of the data. Recognizing such distributions aids in selecting appropriate analytical methods and interpreting the results effectively.
- **Adapting Methodologies:** The data exploration process revealed the importance of adapting data analysis methodologies to suit specific datasets and project goals. Flexibility in methodologies allows for more accurate interpretations and relevant outcomes.

Reflecting on these lessons, we acknowledge the significance of data-driven decision-making and the impact it can have on project success. The exploration of job seekers' and job offers' datasets has paved the way for informed planning and effective implementation in subsequent chapters.

Clarification

As we progress with the project, these lessons will continue to guide us in maximizing the value of the data and optimizing our approach towards achieving the stated project objectives. Embracing a data-driven mindset and continually learning from the data exploration process will foster a more comprehensive understanding of the domain and result in meaningful contributions to the field.

Conclusion and Next Step

In this project, our primary goal is to develop a data-driven solution that leverages advanced recommender systems and artificial intelligence (AI) algorithms to provide personalized job recommendations to job seekers and assist recruiters in finding the best candidates for their job openings. The project entails a comprehensive exploration of two major datasets: the job offers dataset, containing over 60,000 IT job offers with 16 features, and the job seekers' dataset, encompassing diverse information, including job titles, resumes, age, and experience level.

To ensure data quality, we conducted data preprocessing tasks, addressing missing values and normalizing distributions. Additionally, we delved into Natural Language Processing (NLP) preprocessing, handling the extensive textual data within the job descriptions and job seekers' resumes. By adopting the CRISP-DM methodology, we embraced an iterative approach, continuously refining data processing steps. As we progress, each chapter will represent a sprint, covering specific phases like data understanding, data preprocessing, and model development, leading us closer to the ultimate goal of delivering an efficient and personalized job recommendation system. Throughout this journey, we remain committed to employing the best methodologies, leveraging cutting-edge technologies like Python, Talend Open Studio, and PowerBI, and collaborating effectively using Microsoft Teams. Our unwavering dedication to data-driven insights, along with a strong foundation laid through explorations and preprocessing, positions us for success as we embark on the upcoming chapters of our project.

Chapter 3

Data Preparation

Introduction

This chapter will focus on a range of comprehensive data preprocessing steps carefully designed to enhance the data's quality and usefulness. Moreover, given the substantial portion of unstructured textual data within the job descriptions and job seekers' resumes, we recognize the necessity of tapping into the power of Natural Language Processing (NLP) techniques. Leveraging NLP, we will effectively transform this unstructured text data into meaningful features, paving the way for deeper insights and personalized job recommendations.

3.1 Background and Context

Our primary focus is on the essential data-cleaning steps required to extract meaningful information from the datasets. We recognize the significance of data cleaning in ensuring the integrity and reliability of our analyses and modeling tasks. Moreover, as we delve into Natural Language Processing (NLP) tasks, we emphasize the principles and algorithms we employ to effectively handle the unstructured textual data within the job descriptions and job seekers' resumes.

3.1.1 Missing Values

One of our primary tasks is to handle missing values that may exist in the datasets. Fortunately, the missing values in job offers and job seekers' datasets are relatively minimal, comprising less than 1% of the overall data. Although this proportion is not substantial to ensure the best models precision.

It is important to note that the missing values in the datasets pertain to text data, which presents a unique challenge. Unlike numerical data, where techniques like mean imputation or interpolation can be applied to replace missing values, text data requires a different approach.

Replacing text values indiscriminately might risk modifying the data's logical context and introduce unintended biases.

Given these considerations, a prudent approach is to exercise caution while dealing with missing text values. To preserve the inherent logic and patterns within the data, the decision was made to remove the instances with missing values using the "pandas.dropna()" function. By eliminating these missing values, we ensure that the data remains consistent and coherent, preventing any distortion of the original context.
While removing a small percentage of the data might raise concerns about potential information loss, it is important to emphasize that this approach is necessary to maintain the data's fidelity. The minimal impact of missing values on the overall dataset ensures that the removal will not compromise the quality or the validity of our subsequent analyses and modeling tasks.

3.1.2 Data Preparation

A significant portion of our data is text-based, specifically job descriptions and job seekers' resumes. These textual components hold a wealth of valuable information that will be pivotal in the development of our future machine learning algorithms for job recommendations and resume screening.

Therefore, by performing a comprehensive text cleaning that covers all possible cases, we create a pristine textual dataset that will be instrumental in the success of our machine learning algorithms. Effective text cleaning will not only enable us to uncover latent patterns and relevant keywords from job descriptions but also help us understand the essential skills, qualifications, and experiences captured in job seekers' resumes (We will go through the details of each cleaning step in the implementation and execution step).

In contrast, for other numerical attributes like candidates' age and years of experience, normalization might not be necessary due to their inherent normal distribution and limited range of values. The absence of large value discrepancies ensures that these attributes are already in a suitable format for analysis, requiring minimal additional preprocessing. However, we will still perform quality checks and data validation to maintain data integrity throughout the entire dataset.

3.1.3 Text Vectorization

In the realm of machine learning, text data presents a unique challenge as most models are not equipped to directly interpret and process textual information.

And here comes the data vectorization step which is a transformative process that con-

verts text data into numerical vectors, allowing machine learning models to comprehend and analyze textual information effectively. And on this project, I tested TF-IDF (short for term frequency–inverse document frequency) and Word2Vec.
Word2Vec, a word embedding technique, encodes words as continuous vectors in a semantic space. And, on the other hand, TF-IDF is a count-based approach, that calculates the significance of each word within a document relative to its occurrence in other documents.
In the subsequent sections, I will delve into the intricacies of each technique, providing a comprehensive exploration of Word2Vec and TF-IDF.

TF-IDF

TF-IDF[14] (Term Frequency-Inverse Document Frequency)[**?**] is a fundamental text vectorization technique that plays a pivotal role in Natural Language Processing (NLP) and information retrieval tasks. Its significance lies in its ability to capture the importance of words within a document relative to their occurrence across a corpus of documents. In essence, TF-IDF evaluates the relevance of a term to a specific document while considering its occurrence in the entire dataset, making it a powerful tool for feature extraction and text analysis.

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Where:

- t represents a term (word) in the document collection.
- d represents a specific document within the collection.
- D represents the entire document collection.
- TF(t, d) is the number of occurrences of term t in document d. It can be calculated as TF$(t, d) =$ number of occurrences of term t in document d.
- IDF(t, D) is the inverse document frequency of term t in the entire collection of documents D. It can be calculated as IDF$(t, D) = \log\left(\frac{\text{total number of documents in } D}{\text{number of documents containing term } t}\right) + 1$. The "+1" is often added to avoid division by zero when a term does not appear in any document.

In the context of building a personalized job recommendation system, TF-IDF offers several advantages as a text vectorization technique. It effectively extracts essential features from job descriptions and job seekers' resumes, capturing the significance of keywords and phrases, enabling numerical representations suitable for machine learning algorithms.
TF-IDF assigns higher weights to terms that are crucial in describing specific job offers or job seekers' profiles, emphasizing the most relevant information for accurate job matching and recommendations. The sparsity of TF-IDF vectors reduces computational complexity and

memory usage, especially with large datasets, providing an efficient representation. Moreover, TF-IDF's flexibility allows for customization, tailoring the weighting scheme and preprocessing techniques to specific project requirements. However, TF-IDF has limitations, including the

potential loss of context, the lack of consideration for word order and semantic relationships, and its handling of out-of-vocabulary words. These limitations may impact the accuracy and contextual understanding of vectorized representations. To address these shortcomings, exploring advanced methods like word embeddings and combining TF-IDF with other techniques can enhance the accuracy and performance of the personalized job recommendation system.

Word2Vec

Word2Vec[10] is a word embedding technique, that operates on the principle of distributional semantics, which posits that a word's meaning is defined by the context in which it frequently appears. The underlying idea behind Word2Vec is to represent each word in a fixed vocabulary as a numerical vector called a dense vector, capturing the contextual relationships between words within a large corpus of text.
In the Word2Vec approach, we traverse the text corpus, considering each word in the context of its neighboring words. The word at the center (c) is the focus word, and the words surrounding it (o) are the context words(outside word) the number of outside words used for training is fixed through the hyperparameter "window size" which presents the number of surround words in each side of the center word. The key concept of Word2Vec is to calculate the probability of observing a context word (o) given a center word (c)(skip gram[11]) or vice versa. We achieve this by measuring the similarity between the word vectors representing c and o. If two words have similar vectors, the probability of one occurring given the other is high.

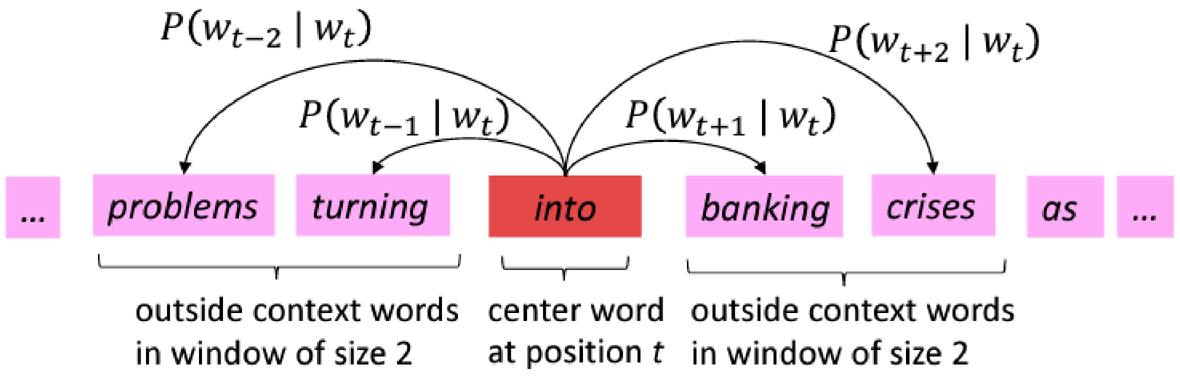


Figure 3.1: Word2Vec Skip-gram

The Word2Vec model iteratively adjusts the word vectors to maximize this probability. Through this iterative training process, the word vectors evolve to effectively capture the underlying semantic relationships in the text corpus. Words that often appear in similar contexts end up having similar vector representations, aligning with their semantic similarity. The like-

lihood function in Word2Vec estimates the probability of context words within a fixed-size window, given the center word W. For each position t=1,. . . ,t, predict context words within a window of fixed size m, given center word W.

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m\leq j\leq m \\ j\neq 0}} P(w_{t+j} \mid w_t; \theta)$$

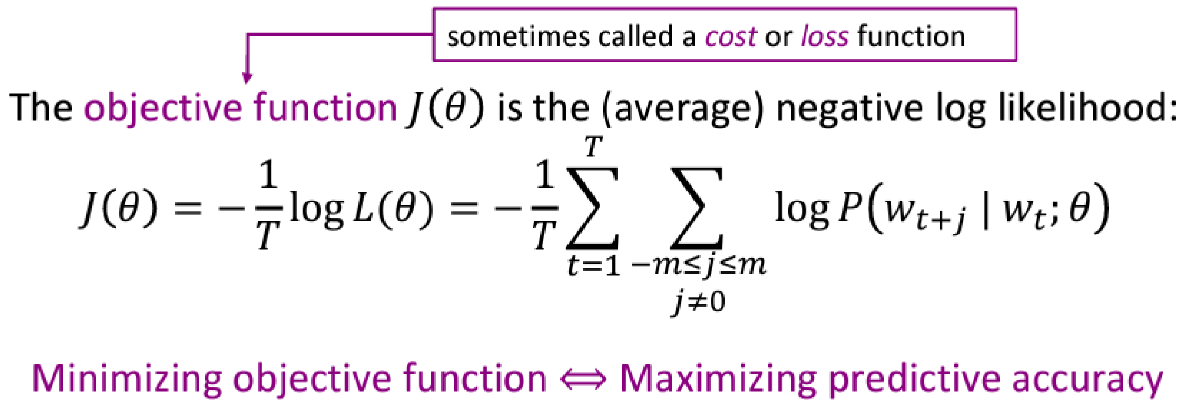
θ is all variables to be optimized

Figure 3.2: Word2Vec Lekelihood Function

where:

- m is the fixed-size window (context window size).
- **W** represents the word vector of the center word.
- T is the total number of positions in the sequence of center words.

The objective of the Skip-gram model is to maximize this likelihood function by learning word embeddings that effectively capture the co-occurrence patterns of words in a given corpus of text. his leads directly to the cost function of Word2Vec, also known as the objective function $J(\theta_i)$, which is based on the (average) negative log-likelihood. The objective is to minimize this cost function during the training process:


sometimes called a *cost* or *loss* function

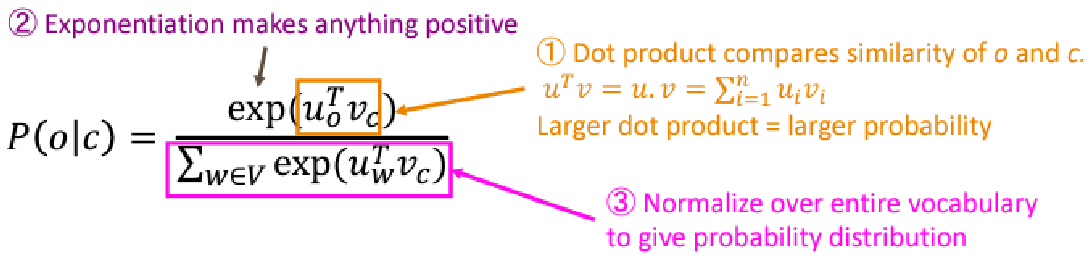
The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log P\left(w_{t+j} \mid w_t;\theta\right)$$

Minimizing objective function ⟺ Maximizing predictive accuracy

Figure 3.3: Word2Vec Objective Function

The challenge in minimizing the objective function lies in calculating the conditional probability $\mathbf{P}(\mathrm{w} \mid \mathrm{w}';\theta)$, which represents the probability of observing a context word given the center word W and the model parameters θ. The equation to compute this probability is known as the Softmax function.


$$P(o|c) = \frac{\exp(\boxed{u_o^T v_c})}{\boxed{\sum_{w\in V}\exp(u_w^T v_c)}}$$

② Exponentiation makes anything positive
① Dot product compares similarity of *o* and *c*.
$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$
Larger dot product = larger probability
③ Normalize over entire vocabulary to give probability distribution

Figure 3.4: Softmax Function

We will use two vectors per word w:
V: when w is a center word.
U: when w is a context word.

The resulting word vectors represent semantic relationships in a dense and continuous vector space. These vectors encapsulate the meaning of words and the contextual patterns they share with other words. Consequently, similar words are positioned close to each other in the vector space, enabling operations like word analogies (e.g., ”king” - ”man” + ”woman” ≈ ”queen”) and measuring word similarity based on vector distances. In other words, in the trained Word2Vec dimensional space, a job description containing data science vocabulary will be closer to a resume with data science-related terms than to a resume focused on web-related skills which will result in a better resume ranking and job recommendations. The proximity between word vectors in the embedding space reflects the semantic similarity between words and their contexts.

Since Word2Vec captures the distributional patterns of words, it can effectively recognize the shared context between data science-related terms in both the job description and the resume, leading to a closer distance between their respective word vectors.

When it comes to Machine learning, Word2Vec uses a three-layer neural network to compute the probability of a word given another word in a given context. The first layer, known as the input layer, represents the one-hot encoded input word, and the second layer, referred to as the hidden layer, holds the word embeddings or distributed representations of words. These embeddings capture the semantic relationships between words based on their co-occurrence patterns in a large corpus. The third layer, the output layer, consists of softmax activation units that generate probabilities for each word in the vocabulary based on its similarity to the input word.

By training the neural network on a vast amount of text data, Word2Vec can learn high-quality word embeddings, enabling efficient representation of words and revealing semantic relationships in the vector space, the image below illustrates the architecture of the neural network used to train wor2vec:

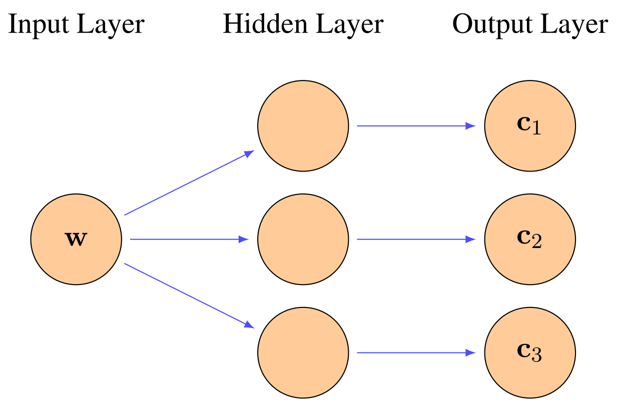


Figure 3.5: Skip-gram Neural Network Architecture

Review

Word2Vec's ability to capture semantic meaning and context has made it a widely adopted technique for various natural language processing tasks. In the context of my project, Word2Vec's word embedding capabilities can enrich the data vectorization process, enhancing the accuracy and effectiveness of our personalized job recommendation system.

3.2 Implementaion and Execution

Here we will delve into the comprehensive text cleaning pipeline implementation, exploring each step in detail to ensure high-quality data preparation. Furthermore, we will concentrate on the implementation of the word2vec transformation technique.

3.2.1 Text Cleaning Pipeline

Below, we will walk through each step of our first-class text-cleaning pipeline:

Removing New Lines Tabs

In this step, we implemented a custom function that utilizes Regular Expressions (RegEx) to effectively remove all occurrences of newlines, tabs, and specific character combinations like \\n and \\ from the input text.

The function `remove_newlines_tags` takes one argument, which is of type `String` and represents the text that requires cleaning and normalization. The function then returns the processed text representing the cleaned version of the original input text, which is also of type `String`.

By employing RegEx patterns, the function efficiently identifies and eliminates newline characters. Here's an example to demonstrate the function's usage:

```
def remove_newlines_tabs(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```
input_text = "This is her \\ first day at work.\n Please,\t Be nice
   to her.\\n"
```

Output:

```
output_text = clean_text(input_text)
print(output_text)
```

The output will be:

```
This is her first day at work. Please, Be nice to her.
```

Striping HTML Tags

In the subsequent step, a custom function was implemented to remove all occurrences of HTML tags from the input text.

The function utilizes Regular Expressions and takes one argument, which is of type `String` and represents the text containing HTML tags that need to be processed.

```
def strip_html_tags(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```
input_text =   This   is a nice place to work. <IMG>
```

Output:

```
output_text = strip_html_tags(input_text)
print(output_text)
```

The output will be:

```
This is a nice place to live.
```

Removing Links

In the following step, another custom function was implemented to remove all occurrences of links from the input text. The function utilizes Regular Expressions and takes one argument, which is of type `String` and represents the text containing links that need to be processed.

```
def remove_links(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```
input_text =   To   know more about this job: example.com visit:
   https://exaple.com//Blogs
```

Output:

```
output_text = remove_links(input_text)
print(output_text)
```

The output will be:

```
To know more about this job:
```

Removing Extra White Space

This function is implemented to remove extra whitespaces from the input text. It takes one argument as input, which is of type texttString and represents the text containing extra whitespaces that needs to be processed. The purpose of this function is to ensure that the textual data is clean and well-formatted.

```
def remove_whitespace(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```
input_text =   Junior        data    scientist          who
   specializes in   machine      learning.
```

Output:

```
output_text = remove_whitespace(input_text)
print(output_text)
```

The output will be:

```
Junior data scientist who specializes in machine learning.
```

Removing Accented Characters

By employing the "Unicode" Python library method called `unicode()`, the function efficiently identifies and removes all accented characters present in the input text, replacing them with their corresponding non-accented versions.

```
def accented_characters_removal(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```
input_text = "MÃ¡laga, Ã  Ã©ÃªÃ¶hello."
```

Output:

```
output_text = accented_characters_removal(input_text)
print(output_text)
```

The output will be:

```
Malaga, aeeohello.
```

Lowercasing Text

In the subsequent step, a custom function was implemented using the built-in Python method `lower()`, which converts the text into lowercase. The function takes one argument, which is of type `String` and represents the text that needs to be converted to lowercase.

```
def lower_casing_text(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```
input_text = "WE are Looking for A Data Scientist."
```

Output:

```
output_text = lower_casing_text(input_text)
print(output_text)
```

The output will be:

```
we are looking for a data scientist.
```

Removing Repeated Characters and Punctuations

This function use `RegEx` to reduce repetitions to two characters for alphabets and to one character for punctuations. If a character is repeated more than two times, it will be reduced to two repetitions for alphabets and one repetition for punctuations.

```
def reducing_incorrect_character_repeatation(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```
input_text = "Realllllllllyyyyy!!!!?"
```

Output:

```
output_text = reducing_incorrect_character_repeatation(input_text)
print(output_text)
```

The output will be:

```
Reallyy!?
```

Expanding Contraction Words

This function expands shortened words to their actual form. For this purpose, a `Python dict` was defined that contains all the shortened forms mapped to their full forms.

The function takes one argument, which is of type `String` and represents the text containing shortened words that need to be expanded.

```python
def expand_contractions(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```python
input_text = "ain't, aren't, can't"
```

Output:

```python
output_text = expand_contractions(input_text)
print(output_text)
```

The output will be:

```
is not, are not, cannot
```

Removing Special Characters

This function uses `RegEx` to remove all special characters that are not required from the input text.
The function takes one argument, which is of type `String` and represents the text containing special characters that need to be removed. The primary objective of this function is to ensure that the textual data is free from unnecessary special characters, maintaining the relevant information and enhancing the readability of the text.

```python
def removing_special_characters(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```python
input_text = "Hello, K-a-j-a-l. Thi*s is $100.05 : the payment that
   you will receive!"
```

Output:

```python
output_text = removing_special_characters(input_text)
print(output_text)
```

The output will be:

```
Hello, Kajal. This is $100.05 : the payment that you will receive!
```

Removing Stopwords

This function removes stopwords, which are words that do not add significant meaning to a sentence and can be safely omitted without compromising the overall meaning of the text.
The function takes one argument, which is of type `String` and represents the text containing words that may be considered stopword.
The primary purpose of this function is to enhance the relevancy and informativeness of the text by eliminating words that are commonly used and do not carry substantial meaning in the context. By using the stopwords list of words provided by the `nltk` library, then the function Converts the text to a list of tokens without stopwords.
The function then returns the processed text as a `list`, representing the tokens extracted from the input.

```
def removing_stopwords(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```
input_text = "This is Kajal from delhi who came here to work."
```

Output:

```
output_text = removing_stopwords(input_text)
print(output_text)
```

The output will be:

```
['Kajal', 'delhi', 'came', 'work', '.']
```

Spelling Correction

This function corrects spellings using `autocorrect.speller` module. The function takes one argument, which is of type ”String” and represents the text containing misspelled words that need to be corrected.
The main objective of this function is to enhance the accuracy and readability of the text by automatically correcting misspelled words.

```
def spelling_correction(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```
input_text = "This is Oberois from Dlhi who came heree to workk."
```

Output:

```
output_text = spelling_correction(input_text)
print(output_text)
```

The output will be:

```
This is Oberoi from Delhi who came here to work.
```

Lemmatization

This function converts words to their lemma or root words. It takes one argument, which is of type `String`. and represents the text containing words in various tenses and plural forms. The main purpose of this function is to simplify the text and ensure that only the root words are retained, eliminating tense forms and plural variations. By using `nltk.stem.WordNetLemmatizer()` module, the function efficiently identifies and transforms words to their base or root form.

```
def lemmatization(input_text):
    # Function implementation goes here
    return cleaned_text
```

Input:

```
input_text = "text reduced."
```

Output:

```
output_text = lemmatization(input_text)
print(output_text)
```

The output will be:

```
text reduce.
```

Review

Our text cleaning pipeline covered various aspects, including removing special characters, handling accented characters, reducing word repetition, expanding shortened words, and removing stopwords. By implementing these steps, we ensured that the data is devoid of irrelevant information, making it more suitable for natural language processing tasks and machine learning algorithms.

3.2.2 Word2Vec Implementaion

`Gensim` library provides an efficient and user-friendly framework for Word2Vec implementation.

To begin, we load the preprocessed text data that has undergone the comprehensive cleaning pipeline we previously discussed. This ensures that the text is in an optimal format for Word2Vec training.

To ensure that our Word2Vec model captures the semantic relationships between words effectively, we decided to combine all the job descriptions and job seekers' resumes into a single corpus. This combined corpus contains approximately 93,000 job seeker resumes and job offer descriptions, and it will be used for trainning the word2vec model.

Combining these two sets of data has several advantages. Firstly, it significantly increases the size of the vocabulary used for training the Word2Vec model. A larger vocabulary provides the model with a more diverse and comprehensive representation of words, resulting in richer word embeddings that capture nuanced semantic relationships.

Furthermore, by merging the job descriptions and job seekers' resumes, we can leverage the shared vocabulary between the two datasets. Job descriptions and resumes within the same field often use similar terminologies and domain-specific language. By incorporating both sources of text data, the Word2Vec model can better capture the specific terminology and context that are relevant to the job market.

Additionally, this approach allows us to learn word embeddings that are not only contextually informed by individual job descriptions or resumes but also by the broader context of the entire corpus. This holistic view enhances the model's ability to capture general patterns and meanings across the entire dataset, resulting in more robust and accurate word embeddings.

Then, we carefully selected hyperparameters to ensure optimal word embeddings that capture meaningful semantic relationships. These hyperparameters play a crucial role in shaping the quality and performance of the Word2Vec model.

Word2Vec Model Parameters

- `Window Size (window)` : We chose a window size of 20 for our Word2Vec model. The window size determines the maximum distance between the target word and the context words within a sentence. A larger window size allows the model to consider a broader context, capturing more contextual information, and potentially better understanding the relationships between words.
- `Dense Vector Size (vector_size)` : We set the dense vector size to 300. This parameter determines the dimensionality of the word embeddings, meaning each word is represented by a vector of 300 dimensions. A higher vector size allows the model to learn more complex and fine-grained word representations, which can be beneficial for capturing intricate semantic nuances in the data.
- `Epochs (epochs)` : We chose to train the model for 20 epochs. An epoch represents the number of times the model goes through the entire dataset during training. Training for multiple epochs allows the model to update its parameters and optimize word embeddings iteratively. However, training for too many epochs may risk overfitting, so finding the right balance is essential.

- `Minimum Word Count (min_count):` We set the minimum word count to 2. This parameter controls the minimum frequency threshold that a word must have to be included in the vocabulary. Words with occurrences below this threshold are discarded from the training process. Setting a minimum count helps filter out rare and potentially noisy words, leading to more meaningful and relevant word embeddings.

Review

By fine-tuning those hyperparameters based on experimentation and evaluation of our specific dataset, we aim to strike the right balance between capturing the semantic relationships among words while avoiding overfitting or memorization of specific patterns in the data. The combination of the chosen hyperparameters allows us to build a robust and informative Word2Vec model that effectively represents the underlying semantics.

3.3 Results and Finding

In this crucial section, we present the outcomes of the comprehensive text cleaning pipeline we meticulously applied to our dataset.

3.3.1 Text Cleaning Pipeline Results

Upon applying the pipeline, we successfully removed newlines, tabs, and unwanted special characters, effectively eliminating distracting elements from the text. HTML tags were efficiently stripped, ensuring that the text remained free of any code artifacts. Additionally, we skillfully handled missing values, avoiding any loss of crucial information while preserving the logic and pattern in the data.

Furthermore, the pipeline expertly tackled the challenges posed by unstructured text. By leveraging advanced NLP techniques, we successfully converted the text into lower case, expanded shortened words, and reduced repetition, enhancing the readability and interpretability of the content. The removal of accented characters and stopwords further refined the text, eliminating irrelevant details without compromising the overall meaning.

After diligently applying our text cleaning pipeline to a resume that contained a substantial amount of noise and an extensive array of words, we obtained impressive results that significantly refined the content. In the screenshot presented below, we showcase the list of words resulting from the text cleaning process, showcasing the most relevant words in their root forms.

Here's a sample of the result that we got:

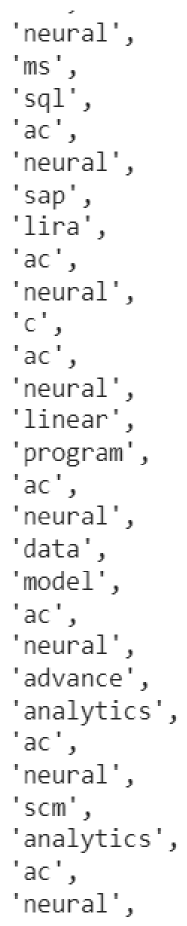

```
'neural',
'ms',
'sql',
'ac',
'neural',
'sap',
'lira',
'ac',
'neural',
'c',
'ac',
'neural',
'linear',
'program',
'ac',
'neural',
'data',
'model',
'ac',
'neural',
'advance',
'analytics',
'ac',
'neural',
'scm',
'analytics',
'ac',
'neural',
```

Figure 3.6: Text Cleaning Results

As depicted in the screenshot, the text cleaning pipeline effectively removed unnecessary noise and punctuation while successfully reducing words to their lemma or root forms. This transformation allowed us to focus solely on the most essential keywords and relevant information in the resume, thereby enhancing its readability and simplifying subsequent data processing.

The refined list of words captures the candidate's core expertise, showcasing valuable skills like ”neural,” ”linear,” and ”analytics.” Moreover, the pipeline preserved critical technical terms like ”sql,” ”sap,” and ”program,” which play a pivotal role in assessing the candidate's qualifications.

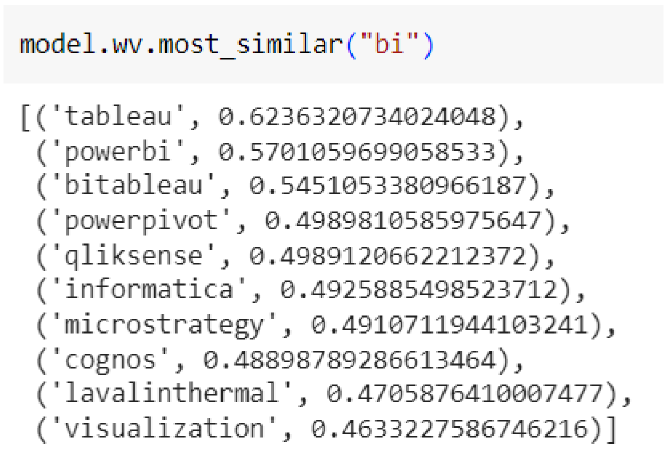
Overall, the results obtained after applying the text cleaning pipeline demonstrate its prowess in transforming complex and noisy text data into a structured and concise representation, empowering our system to make accurate and meaningful recommendations to both job seekers and recruiters.

3.3.2 Word2Vec Results

Word2Vec implementation has transformed each word into a rich data point represented by a dense 300-dimensional vector. This vector encapsulates the semantic meaning and contextual information of the word, capturing its relationships with other words within the dataset. The resulting vectors hold both positive and negative values, effectively encoding the nuances and associations of the language.

As a result, every unique word in the vocabulary is now represented by a unique vector, forming a comprehensive dictionary where each word serves as a key, and its corresponding vector serves as the associated value. This dictionary becomes a valuable resource that we can utilize throughout our project for various NLP tasks and machine learning algorithms.

We can showcase its effectiveness by calculating the most similar words to a given word. In this case, we chose the word ”bi” and utilized the trained model to find the words that are closest to ”bi” in the 300-dimensional space, based on cosine similarity.



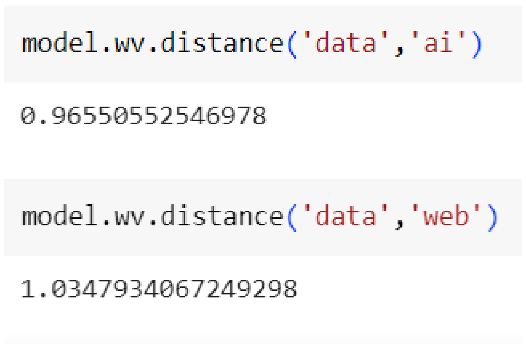
```
model.wv.most_similar("bi")
```

```
[('tableau', 0.6236320734024048),
 ('powerbi', 0.5701059699058533),
 ('bitableau', 0.5451053380966187),
 ('powerpivot', 0.4989810585975647),
 ('qliksense', 0.4989120662212372),
 ('informatica', 0.4925885498523712),
 ('microstrategy', 0.4910711944103241),
 ('cognos', 0.48898789286613464),
 ('lavalinthermal', 0.4705876410007477),
 ('visualization', 0.4633227586746216)]
```

Figure 3.7: Word2Vec Results: Similar Words to the word bi

The obtained screenshot reveals the results of this process. The model returns a list of words that exhibit high cosine similarity with the word ”bi,” demonstrating words that are semantically similar or conceptually related to ”bi.” These similar words might include terms like ”tableau,” ”powerbi,” ”qliksense,” ”visualization,” and others that share similar themes or contexts with the original word ”bi.”

we examined also the distance between certain words in the learned word embeddings. Specifically, we calculated the cosine similarity between the word vectors of ”data,” ”AI,” and ”web.” The cosine similarity score reflects the semantic similarity between words, with higher values indicating closer semantic relationships.



```
model.wv.distance('data','ai')
0.96550552546978
model.wv.distance('data','web')
1.0347934067249298
```

Figure 3.8: Word2Vec Result: Word distance

3.4 Lesson Learned and Reflection

In retrospect, our text cleaning pipeline has been a pivotal aspect of our project, and it has taught us valuable lessons about dealing with complex and noisy text data. Its comprehensiveness in handling various cases proved to be both beneficial and challenging.

On the positive side, the text cleaning pipeline has significantly improved the quality of our data. By removing unwanted characters, links, and html tags, we were able to obtain cleaner and more structured text. Additionally, the elimination of special characters and stopwords contributed to a more focused and meaningful representation of the data, enhancing the effectiveness of subsequent analyses. Furthermore, the expansion of shortened words to their full forms and the correction of misspelled words using autocorrect.speller have not only improved the readability of the text but also laid a solid foundation for accurate language processing.

However, on the flip side, the comprehensive nature of the pipeline has introduced complexities in terms of computational resources. As our data is large and contains significant amounts of text, processing the entire dataset with our text cleaning pipeline requires substantial computing power and time. This poses a challenge in terms of scalability and efficiency, particularly when dealing with real-time data processing. Nonetheless, we recognize that striking a balance between comprehensiveness and efficiency is crucial. To address this, we may explore optimizing our text cleaning pipeline by employing more efficient algorithms or implementing parallel processing techniques to reduce computational burdens. Overall, our experience with the text cleaning pipeline has taught us the importance of addressing data quality early in the data preprocessing phase. While its complexity can pose challenges, the benefits of cleaner and more meaningful data significantly impact the performance and accuracy of our subsequent machine learning models and algorithms.

Word2vec has indeed demonstrated its remarkable power in capturing the meaning of words and representing them efficiently in a lower-dimensional space through a simple 3-layer neural network architecture. The beauty of word2vec lies in its ability to create dense and continuous word embeddings that preserve semantic relationships between words.

Unlike the traditional TF-IDF vectorization, which produces high-dimensional sparse vectors, word2vec generates dense representations with a fixed number of dimensions (e.g., 300). These dense embeddings contain valuable information about the context and semantic meaning of words, making them ideal for further natural language processing tasks, such as recommendation systems and sentiment analysis. One of the significant advantages of word2vec over TF-IDF in our case is the ability to capture word similarities and semantic relationships. In word2vec, words with similar meanings or used in similar contexts tend to have vectors that are closer to each other in the embedding space. This enables us to measure the semantic similarity between words based on their vector representations. This is particularly valuable in our job recommendation system, as it allows us to find job offers that are closely related to a job seeker's skills and preferences based on the similarity of their word embeddings.

Moreover, word2vec's dense and compact embeddings result in a more computationally efficient representation compared to the large and sparse TF-IDF vectors. The reduction in dimensionality not only speeds up computation but also reduces the memory required to store the word representations, making it more scalable for handling large datasets and real-time applications.

Conclusion and Next Step

In conclusion, the data preprocessing phase has played a pivotal role in preparing our textual data and making it ready for further analysis and modeling. Through a comprehensive text cleaning pipeline, we successfully tackled the challenge of dealing with noisy and unstructured text data, resulting in cleaner and more meaningful representations of job offers and job seekers' resumes. Additionally, we leveraged the power of word2vec to create dense and meaningful word embeddings, effectively capturing the semantic relationships between words. These embeddings form the foundation for our recommender systems and resume screening AI system, enabling accurate job recommendations and resume ranking based on the similarity of word representations.

In the upcoming modeling phase, we acknowledge the significance of this critical sprint, demanding extra time and effort. Building robust recommender systems and AI-based resume screening and ranking systems is challenging but essential for project success. During this sprint, we will focus on developing and fine-tuning machine learning models using word2vec's word embeddings. These models will recommend relevant job offers to seekers and screen resumes based on job requirements and employer preferences.

Chapter 4

Recommender Engine

Introduction

Welcome to the Recommender Engine chapter, where we explore recommendation algorithms for personalized job matching. We'll discuss principles, implement algorithms, showcase results, and reflect on the lessons learned.

4.1 Background and Context

We will explore the various types of recommendation systems, such as collaborative filtering and content-based filtering.

4.1.1 Recommendation Architecture

In our recommendation system, we will implement two distinct architectures for each type of recommendation: the TopN architecture and the K-Nearest Neighbors-based rating prediction (KNN). Initially, the TopN architecture will be employed to provide users with a list of top job offers that match their preferences and interests.
Subsequently, we will delve deeper into our data and explore the KNN algorithm to predict personalized ratings for job offers that users have not yet encountered. By combining these two approaches, we aim to deliver precise and targeted job recommendations, ultimately enhancing the user experience and facilitating more informed and satisfying job search journeys for our users.

TopN Architecture

The figure below illustrates the TopN architecture of our recommendation system. It shows the flow of data and steps involved in generating personalized job recommendations:

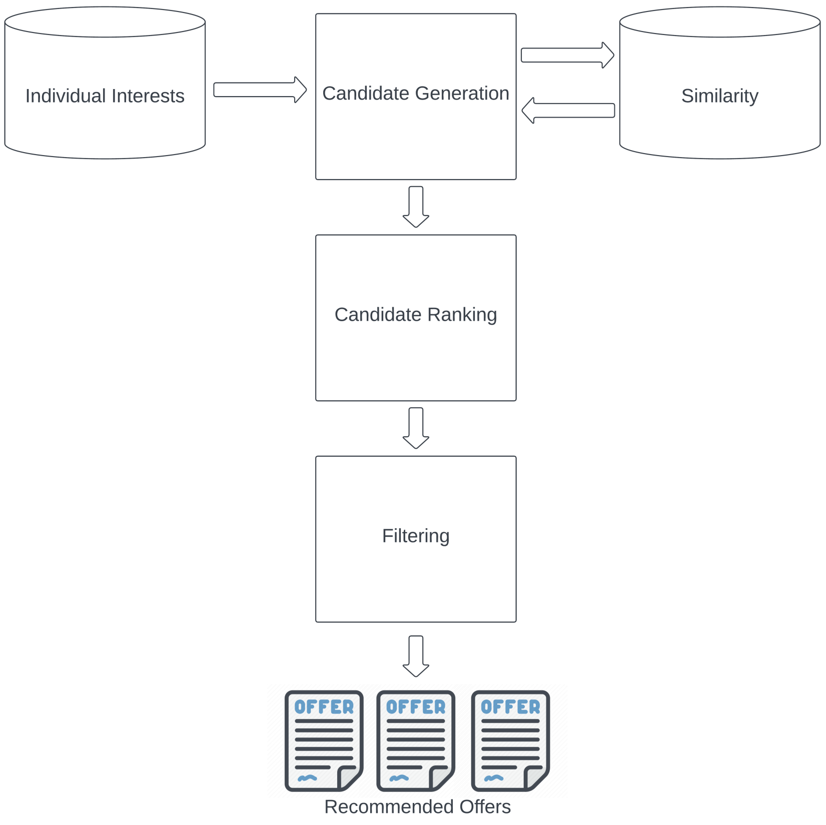


Figure 4.1: TopN Recommendation Architecture

The TopN architecture of our recommendation system operates as follows: We begin with individual data representing job seekers, which includes their ratings for other jobs or implicit data such as job applications they've submitted. The first step is to generate recommendation candidates, which are job offers we believe might be interesting to the user (job seeker) based on their past behavior. This is achieved by computing the similarity between the user's preferences and the attributes of different job offers.

Once we have the recommendation candidates, we proceed to rank them according to their relevance to the user. Alternatively, we can apply a threshold, such as considering only job offers with a rating of 4/5 or higher. After ranking the candidates, we may observe that certain job offers appear multiple times due to their high similarity to the user's preferences. In such cases, we employ filtering techniques to ensure diverse and relevant job offers are presented to the user, eliminating redundancies and improving the overall recommendation quality. This process ensures that job seekers receive tailored job recommendations that align closely with their preferences, leading to a more engaging and

K-Nearest Neighbors Architecture

When employing the K-Nearest Neighbors[19](KNN) algorithm for rating prediction in a recommender system, the process involves taking two essential input parameters: the specific job seeker for whom the recommendation is intended, and a job offer that the job seeker has not encountered previously. The goal is to predict the rating that the job seeker is likely to assign to the unseen job offer. The KNN algorithm operates based on the concept of finding the ”nearest neighbors” of the target job seeker among other users in the system who have similar preferences and historical interactions with job offers.

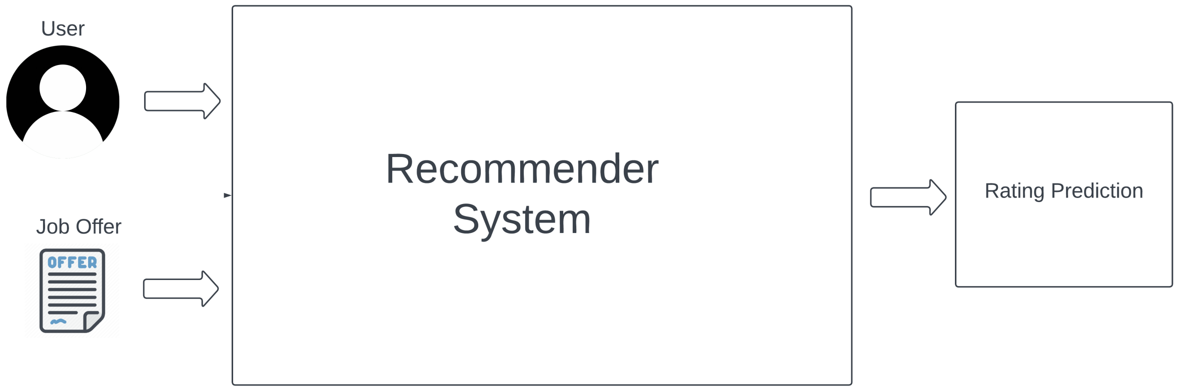


Figure 4.2: Rating Prediction Architecture

The KNN-based recommender system’s architecture can be summarized as follows: It starts by creating a similarity matrix, quantifying job seekers’ similarity based on past interactions with job offers. This matrix measures how closely users’ preferences align. To recommend a new job offer, the system identifies the ”k” nearest neighbors to the target user from the similarity matrix, representing users with similar interests and behavior.

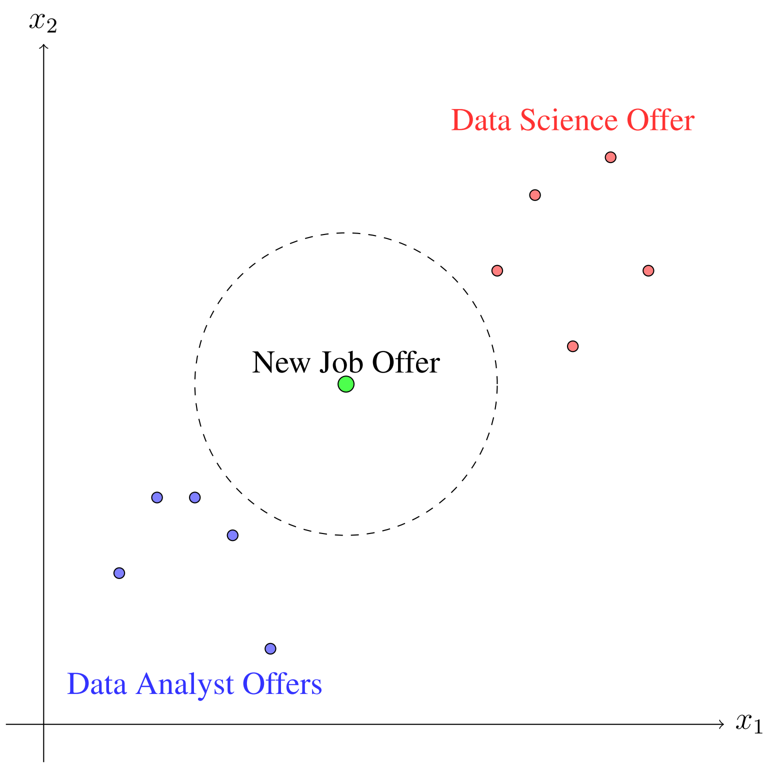


Figure 4.3: Illustration of K-Nearest Neighbors Algorithm (KNN)

Review

Predicting ratings using the KNN approach tends to be more accurate since it considers a larger amount of information from the data, taking into account the similarities between users and their past ratings. On the other hand, the TopN recommendation[5] method might be less accurate but is more straightforward and computationally efficient. It focuses on generating a list of top recommendations for users based on content-based filtering or simpler collaborative filtering approaches, this allows for faster and more scalable recommendations, making it suitable for systems with larger datasets or real-time applications where speed is crucial.

In summary, while the KNN-based approach provides higher accuracy, the TopN method offers a balance between simplicity and efficiency, making it a valuable option for certain recommendation scenarios.

4.1.2 Content-Based Filtering

Content-based filtering[12] is aptly named because it relies on the content of the items being recommended in our case the content of the job offer. By analyzing the attributes and characteristics of each job offer, the system can match job seeker preferences with relevant content, providing personalized recommendations based on user interests and preferences.

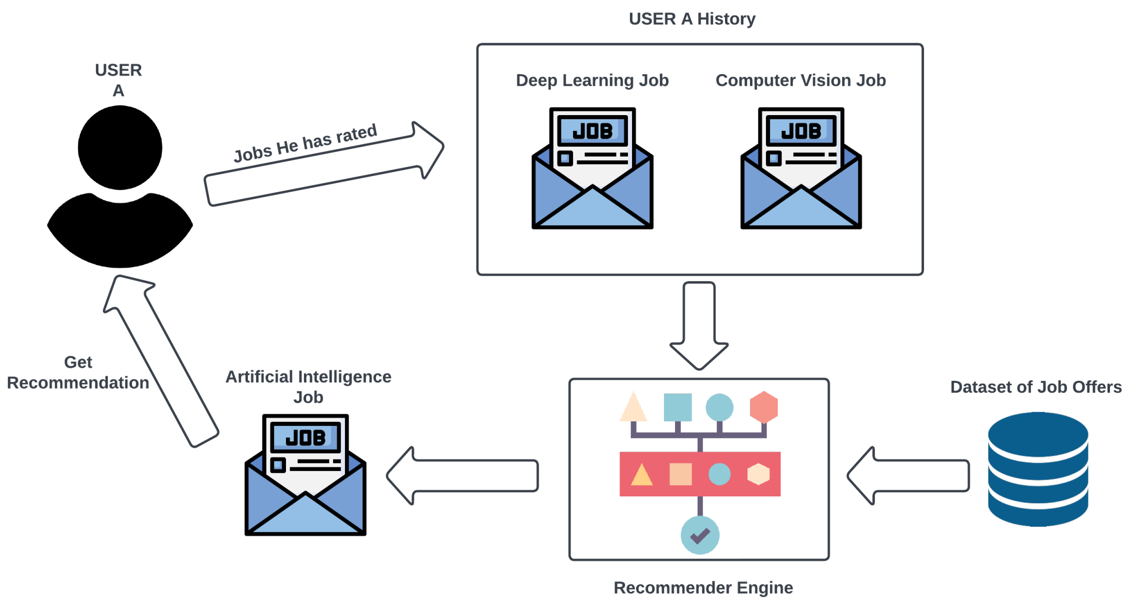


Figure 4.4: Content Based Filtering

In our case, the attributes used for content-based filtering are the job offer description, years of experience required, and salary. These factors are crucial in tailoring recommendations to each user, as they provide valuable insights into individual preferences, professional background, and career expectations.

How It Works?

This method revolves around making job recommendations solely based on the intrinsic attributes of the job offers themselves, as provided by our comprehensive dataset. Within this dataset, we have access to crucial information such as the required experience for each offer, the corresponding salary, and detailed job offer text descriptions.

By tapping into this wealth of data, we can personalize the job-seeking journey for each user. Let's consider a scenario where a job seeker explicitly expresses a preference for job offers that mandate 5 years of experience, boast a salary exceeding 100k USD annually, and are tailored to the domain of data science. Armed with this valuable insight, we are well-positioned to delve into the process of recommending other job offers that bear a striking resemblance to the ones already preferred by the job seeker.

The underlying principle of this content-based filtering mechanism revolves around calculating the similarity between the user's profile, which encompasses their desired job offer attributes, and the attributes of each job offer in the dataset. This similarity analysis enables us to identify job offers that align closely with the job seeker's unique interests and requirements. As a result, the recommendations provided are tailored to each user's specific preferences, thereby significantly enhancing the likelihood of discovering the most suitable and engaging job opportunities.

To bolster the effectiveness and accuracy of our recommender engine, we leverage a multi-step approach. We begin by implementing a rigorous text cleaning pipeline to prepare the raw job offer descriptions for further analysis. We then harness the power of word2vec, which maps words into a dense vector space to capture the semantic meaning and relationships among words. This embedding technique ensures that job offers with similar content are represented closely in the vector space, facilitating more precise and relevant recommendations.

By combining the potency of content-based filtering with our advanced text processing methodologies, we pave the way for an unparalleled job-seeking experience. Our data-driven approach enables us to empower job seekers with personalized, high-quality job recommendations that align closely with their preferences and expertise. Moreover, by employing content-based filtering, we effectively address the need for tailored job matching and offer an intelligent solution to optimize the job search process. As we move forward in this thesis, we will demonstrate how this approach elevates the efficiency and effectiveness of job recommendation systems, ultimately revolutionizing the job-seeking landscape for users worldwide.

Computing Similarity

I employed two similarity metrics. Firstly, I applied cosine similarity to measure textual similarity between job offers and resumes. Secondly, I computed similarity based on years of experience and salary. This combined approach enhances the precision of our recommender systems and resume screening, ensuring more accurate and relevant recommendations.

Cosine Similarity

In the pursuit of calculating the similarity between job offers description texts, we employ a powerful and versatile metric known as cosine similarity[13]. This approach has proven to be highly effective in a wide array of scenarios, making it an ideal choice for our content-based filtering recommendation system.

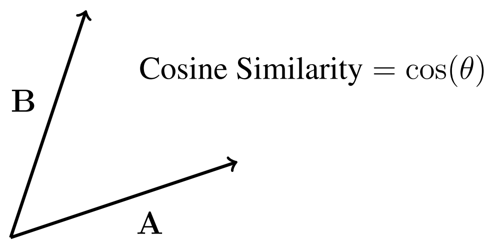


Figure 4.5: Cosine Similarity Illustration

The essence of cosine similarity lies in measuring the angle between two vectors in a multi-dimensional space, such as the vector representations of job offers based on their attributes. By calculating the cosine of this angle, we can determine the similarity between the job offers, providing valuable insights into which offers align most closely with each other.

The values obtained from calculating cosine similarity range between -1 and 1, where -1 signifies completely opposite vectors, 0 indicates orthogonality or independence, and 1 denotes identical vectors. In our case, a cosine similarity value closer to 1 indicates a high degree of similarity between two job offers, implying that they share similar characteristics and would likely appeal to the same set of job seekers. On the other hand, a value closer to -1 denotes dissimilarity between two job offers, indicating that they target different skill sets or interests.

We will compute Cosine Similarity to measure textual similarity between the job offer description the and job seeker's resume after transforming these texts into vectors using word2vec.

The cosine similarity between two vectors X and Y in a n-dimensional space is given by:

$$\text{Cosine Similarity (X, Y)} = \frac{\sum_{i=1}^{n} X_i \times Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2} \times \sqrt{\sum_{i=1}^{n} Y_i^2}}$$

where:

- X and Y are n-dimensional vectors representing two data points in the multidimensional space.
- X_i and Y_i represent the i-th elements of vectors X and Y respectively.

.

Cosine Similarity Clarification

The summation $\sum_{i=1}^{n} X_i \times Y_i$ calculates the dot product between vectors X and Y, which is the sum of the element-wise product of corresponding elements of the vectors. $\sqrt{\sum_{i=1}^{n} X_i^2}$ and $\sqrt{\sum_{i=1}^{n} Y_i^2}$ represent the Euclidean norms (lengths) of vectors X and Y respectively. The Euclidean norm of a vector is calculated as the square root of the sum of the squares of its components.

Salary and Years of Experience Similarity

In assigning a similarity score based on the salary and required years of experience for job offers, there's an element of art involved in designing the recommender system By setting a reasonable starting point, such as a 3-year difference in required experience, we can ensure that job offers with notable distinctions are appropriately accounted for in the recommendation process. For instance, job offers requiring 1 year of experience will be treated as significantly different from those demanding 5 years of experience.

The mathematical function we employ acts as a scaling mechanism, mapping the absolute difference in required experience to a range between 0 and 1. At a difference of 0 years, the similarity score reaches its highest value of 1, indicating complete similarity between the job offers. As the difference increases, the similarity score decreases exponentially, signifying a diminishing similarity between the job offers. Around the 3-year difference mark, the similarity score significantly reduces, indicating that job offers with this degree of disparity are considered less similar. Beyond a difference of 6 years, the similarity score diminishes to nearly zero, highlighting that job offers with substantial discrepancies in required experience are perceived as very dissimilar.

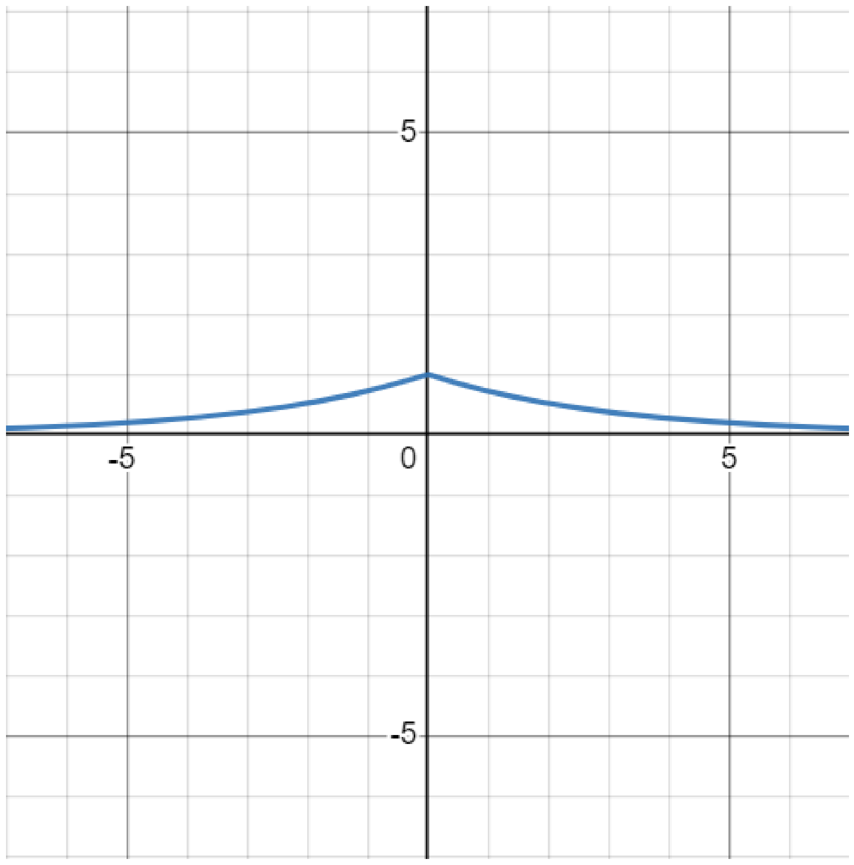


Figure 4.6: Experience Similarity Graph

The equation of the function is given by:

$$f(x_1, x_2) = \exp\left(\frac{|x_1 - x_2|}{3}\right)$$

where:

- x_1 represents the job seeker's years of experience.
- x_2 represents the job offer's required years of experience.

Similarly, we apply a comparable approach to the salary attribute in our content-based filtering algorithm. In this case, we consider a difference of $10,000 USD per year as the threshold for substantial dissimilarity between job offers. Offers that vary in salary by up to $10,000 USD per year are treated as relatively similar, while those with larger salary disparities are deemed less alike.
Let x_1 be the job seeker's expected salary, and x_2 be the job offer's offered salary. The function $f(x_1, x_2)$ is defined as follows:

$$f(x_1, x_2) = \exp\left(\frac{|x_1 - x_2|}{10}\right)$$

TopN Recommendation for Content-Based Filtering

The TopN architecture for content-based filtering, which operates based on the principles we discussed earlier. However, in this context, the candidate generating process is different from the collaborative filtering approach. Instead of relying on similarities between users, we focus on the attributes of the jobs that the user has already shown interest in, such as jobs they have liked or applied for.

The first step in the content-based filtering approach is to extract relevant attributes from the job offers that the user has interacted with positively. These attributes may include the required years of experience, expected salary, and the content of the job description.

Next, using these attributes, we compute the similarity between the user's preferred job offers and the remaining job opportunities in the dataset. This similarity computation helps identify other job offers that share similar attributes and characteristics with the ones the user has shown interest in.

K-Nearest Neighbors for Content-Based Filtering

Once we have the similarity scores, we proceed to rank the job offers based on their relevance to the user's preferences. The top-ranked candidates are then presented as personalized recommendations to the user, aligning closely with their past interactions and preferences.

The challenge now lies in predicting the rating of a job offer that a user has not seen before. To accomplish this, we employ a technique called ”k nearest neighbors” (k-NN). Despite its seemingly fancy name, k-NN is a straightforward and effective concept. We begin by measuring the content-based similarity between all the job offers that a given user has rated and the offer for which we want to predict a rating.

Next, we choose a number, denoted as K, which represents the count of nearest neighbors to the offer we seek to predict a rating for. The term ”nearest” can be defined based on the highest content-based similarity scores to the target offer. In our case, we select the K job offers that exhibit the closest resemblance to the offer we are evaluating for this particular user. For instance, we could choose the top 40 offers that most closely match the one we want to predict.

And that's essentially the concept of ”k nearest neighbors.” It revolves around selecting a set of items that are close to the item of interest, referred to as its ”neighbors,” and making predictions about that item based on the properties of its neighbors.

To convert these top 40 closest offers into an actual rating prediction, we calculate a weighted average of their similarity scores concerning the target offer. These similarity scores are weighted by the ratings the user has assigned to each of the neighboring offers. This way, we derive a prediction for the rating of the job offer the user has not seen before, utilizing the collective information from similar offers that the user has rated. This k-NN approach enables us to provide personalized and relevant job offer recommendations, tailored to the preferences and interests of each individual user.

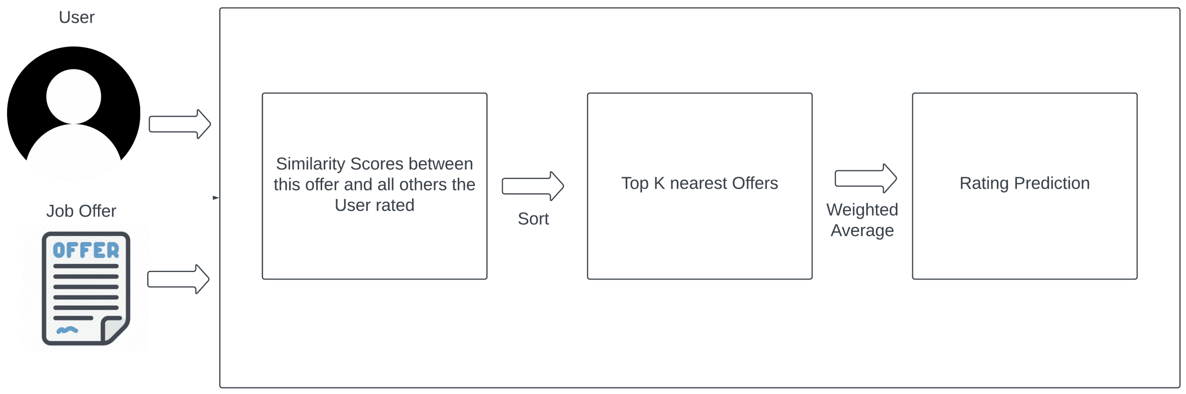


Figure 4.7: Content Based Filtering

4.1.3 Implicit and Explicit Rating

Notice

In our recommendation system project, we primarily focus on explicit ratings, where users explicitly provide feedback by giving job offers a numerical rating. However, we also acknowledge that there are other forms of implicit ratings that can provide valuable insights into a user's preferences and interests.

Implicit ratings occur when users interact with the system in ways that indicate interest, such as clicking on a link or applying for a job. These implicit signals offer crucial information about user behavior, even though they don't provide explicit numerical ratings.

To make our recommendation system more effective and comprehensive, we recognize the importance of incorporating both explicit and implicit signals. By considering implicit feedback, we can capture additional dimensions of user engagement and better understand their preferences. For example, if a user frequently clicks on job offers related to data science, it suggests a strong interest in that field, even if they haven't explicitly rated those offers. Incorporating such implicit signals allows us to offer personalized recommendations that align with a user's actual interests and interactions.

4.1.4 Neighborhood-Based Collaborative Filtering

In this section, we delve into neighborhood-based collaborative filtering[1], a powerful concept that leverages the behavior of others to enhance your job offer recommendations. At its core, it involves seeking out individuals similar to you, or jobs similar to the ones you prefer. This process, often referred to as finding your ”neighborhood,” allows us to recommend job offers based on the preferences of others like you, guiding you towards opportunities that you haven't explored yet. Collaborative filtering embodies the idea of collective collaboration, where insights from individuals with similar interests contribute to providing tailored and relevant job recommendations for your unique preferences.

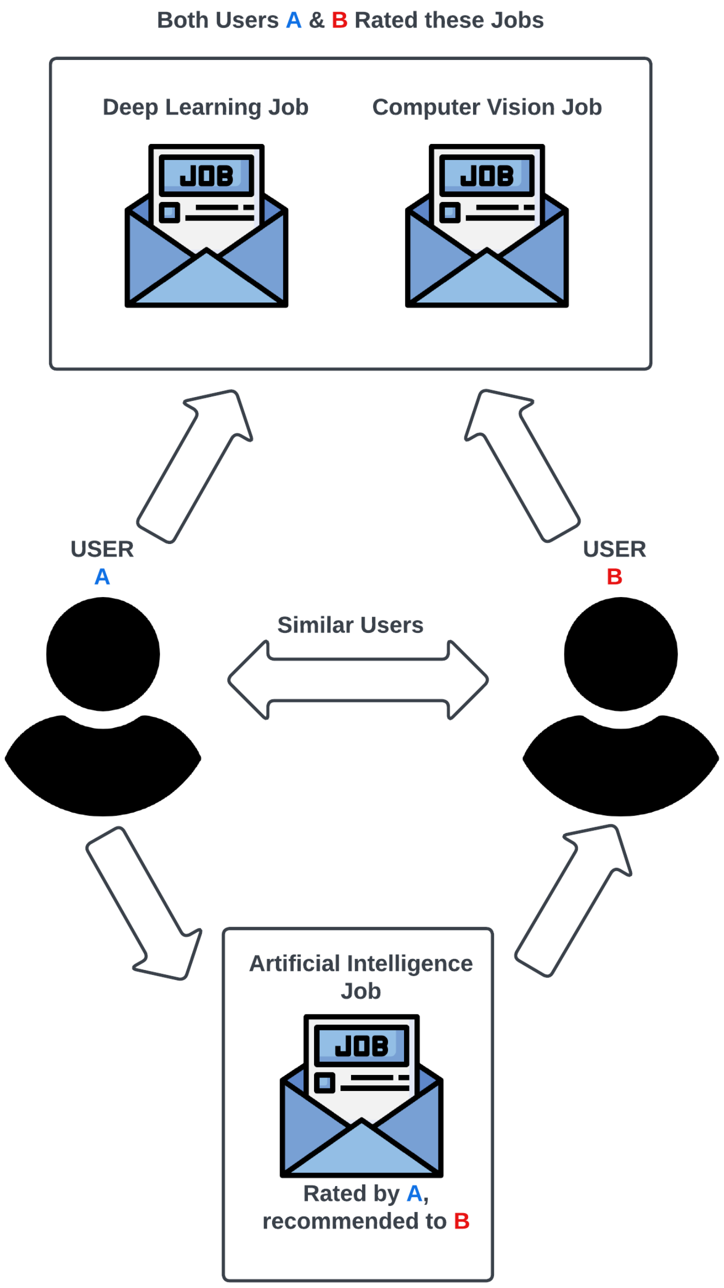

Both Users A & B Rated these Jobs
Deep Learning Job
Computer Vision Job
JOB
JOB
USER A
USER B
Similar Users
Artificial Intelligence Job
JOB
Rated by A, recommended to B

Figure 4.8: Collaborative-Based Filtering

Similarity Computing

The cosine similarity metric is a powerful tool for measuring similarities between vectors, and it generally performs well in many cases. However, to achieve even better results, we can take a step further and delve deeper into its optimization. Adjusting the cosine similarity metric can lead to more accurate and meaningful recommendations, tailoring them to the specific preferences and behaviors of job seekers.

One significant challenge we encounter in using behavior data to measure similarities is the inherent sparsity of the data. With countless job offers available worldwide, it becomes highly improbable that any individual job seeker has encountered every single job opportunity. As a result, when we calculate the similarity between two job vectors, they often contain numerous zeros, representing jobs that the user has not interacted with.

Handling such sparse data poses computational challenges, as it requires substantial computational power to process and compare vectors with numerous zero elements. To address this, we are going to explore a technique to optimize the computation and minimize the computational burden.

Sparse Arrays

To tackle the challenge of sparse data, we will employ a data structure called sparse arrays. Sparse arrays are specifically designed to handle datasets with a large number of zero elements efficiently. Unlike regular arrays, which store all elements regardless of their value, sparse arrays only store non-zero elements, significantly reducing memory usage and computational complexity.

In our context, sparse arrays will be utilized to represent the job offers and job seekers' interactions. Instead of storing all interactions, we will only store the non-zero values, which correspond to the job offers a user has engaged with, such as ratings, applications, or other implicit indications of interest. By doing so, we can transform the original dense data matrix into a more compact and memory-efficient representation, reducing the computational burden associated with sparse data.

Adjusted Cosine

The adjusted cosine metric is a valuable tool, particularly when measuring the similarity between users based on their ratings. It acknowledges that people may have varying rating scales or baselines. For instance, what Bob considers a 3-star offer might differ from what Alice considers a 3-star offer due to their rating tendencies. Bob may be conservative with 5-star ratings, reserving them for truly outstanding offers, while Alice may generously give 5 stars unless she strongly dislikes something. This effect is not only noticeable among different individuals but also across various cultures, where some countries may be more stringent with their ratings.

To address these differences, the adjusted cosine metric normalizes the ratings. Instead of calculating similarities based on raw rating values, it measures similarity by considering the difference between a user's rating for an item and their average rating for all items.

$$s_{ij} = \frac{\sum_u (R_{ui} - \bar{R}_u)(R_{uj} - \bar{R}_u)}{\sqrt{\sum_u (R_{ui} - \bar{R}_u)^2 \sum_u (R_{uj} - \bar{R}_u)^2}} \tag{4.1}$$

Where:

- R_{ui} is the rating given by user u to item i.
- R_{uj} is the rating given by user u to item j.
- $\bar{R}_u$ is the mean rating given by user u to all rated items, calculated as $\frac{1}{N_u} \sum_{k \in I_u} R_{uk}$, where N_u is the number of items rated by user u, and I_u is the set of items rated by user u.
- The sum $\sum_u$ is taken over all users who have rated both items i and j. In other words, it includes only users who have rated both item i and item j.

The adjusted cosine similarity ranges from -1 to 1, where -1 indicates a perfect negative similarity, 1 indicates a perfect positive similarity, and 0 indicates no similarity between the two items.

Pearson Similarity

An intriguing variation of the adjusted cosine metric is the Pearson similarity, which introduces a slight twist. Instead of focusing on the difference between ratings and an individual user's average rating, it examines the difference between ratings and the average rating of a given item across all users.

With Pearson similarity, we no longer attempt to account for the unique personal definitions of specific rating scores for each individual. In real-world scenarios with sparse data, this approach proves beneficial. Pearson similarity can be understood as a measure of similarity between users based on how much their behaviors deviate from the average behavior of all users. It takes into account the variations from the collective norm, allowing us to evaluate the similarity between users in a more robust manner. By considering the collective perspective, Pearson similarity provides a meaningful insight into the shared preferences and tendencies among users, making it a valuable tool for recommendation systems.

$$r_{XY} = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2 \sum_{i=1}^{n} (Y_i - \bar{Y})^2}} \tag{4.2}$$

Where:

- n is the number of data points in the datasets X and Y.

- X_i is the i-th value in dataset X.
- Y_i is the i-th value in dataset Y.
- $\bar{X}$ is the mean of dataset X, calculated as $\frac{1}{n}\sum_{i=1}^{n} X_i$.
- $\bar{Y}$ is the mean of dataset Y, calculated as $\frac{1}{n}\sum_{i=1}^{n} Y_i$.

The Pearson similarity coefficient ranges from -1 to 1, where -1 indicates a perfect negative linear correlation, 1 indicates a perfect positive linear correlation, and 0 indicates no linear correlation between the two variables.

Spearman Rank Correlation

Let's also touch upon Spearman rank correlation for the sake of completeness. Conceptually, it follows a similar path as Pearson similarity, but with a noteworthy difference in the data representation. Instead of directly using rating scores, we work with ranks. This means that instead of relying on the average rating value for a job offer, we utilize its rank among all job offers based on their average ratings. Similarly, instead of individual ratings for a job offer, we consider the rank of that job offer among all the individual's ratings.

By employing ranks instead of raw ratings, Spearman rank correlation allows us to measure the similarity between users or job offers in a more robust and non-parametric way. This approach is particularly useful when dealing with sparse data or when the underlying distribution of ratings is not normally distributed. The use of ranks helps mitigate the influence of outliers and provides a more resilient similarity metric, making it a valuable addition to our arsenal of recommendation techniques.

Mean Squared Difference

Another approach to measuring similarity is the mean squared difference similarity metric, which is as straightforward as it sounds. To compute this metric, we consider all the job offers that two users have in common in their ratings. For each shared job offer, we calculate the squared difference between how each user rated it. Then, we take the mean of these squared differences.

This method is relatively easier to grasp compared to cosine similarity, as it doesn't involve angles in multidimensional space. Instead, it directly compares how two individuals rated the same set of job offers. It's akin to how we measure mean absolute error when evaluating the overall accuracy of a recommender system.

$$\mathrm{MSD}_{ij} = \frac{1}{N_{ij}} \sum_{u} (R_{ui} - R_{uj})^2 \tag{4.3}$$

Where:

- R_{ui} is the rating given by user u to item i.
- R_{uj} is the rating given by user u to item j.
- The sum $\sum_u$ is taken over all users who have rated both items i and j. In other words, it includes only users who have rated both item i and item j.
- N_{ij} is the number of users who have rated both items i and j. It is the normalization factor to ensure fairness when comparing items with different numbers of user ratings.

The MSD ranges from 0 to a positive value, where 0 indicates that the items have identical ratings by all users who rated them, and a higher value indicates greater dissimilarity between the two items.

User-Based Collaborative Filtering

Among the various approaches, user-based collaborative filtering stands out as the most straightforward and intuitive, making it an excellent starting point for our exploration.

TopN Recommendations

The concept underlying user-based collaborative filtering is quite straightforward. First, identify other job seekers who exhibit similar rating patterns to yours based on their history of ratings. Then, recommend job offers that these similar job seekers liked but you haven't seen yet.

The initial step involves gathering the necessary data, which consists of a table containing all the ratings from individuals within our system. This can be visualized as a 2D array with job offers on one axis, job seekers on the other, and rating values in each cell.

Table 4.1: User-Offer Rating Matrix

	Data Engineer	Data Analyst	Data Science	Cloud Computing
Bob	5	0	0	0
Ted	0	5	5	4
Ann	4	5	0	0

For instance, Ann expressed appreciation for both a data engineer offer and a data analyst offer, while Bob only saw and liked the data engineer offer. Additionally, they each encountered other offers that they don't have in common. Based on this information, it seems probable that Bob might also enjoy the data analyst offer that Ann rated, given their shared interest in the data engineer offer.

Visualizing this data, we can perceive it as representing vectors for each job seeker in the item space. Suppose these were the only five job offers available; then we could describe Bob with a 4-dimensional vector of (5, 0, 0, 0) and Ann with (4, 5, 0, 0), wherein 0 denotes a missing value. This setup provides the necessary foundation to calculate the cosine similarity score between any two users. Additionally, we have the flexibility to experiment with various similarity metrics as well.

Table 4.2: User Similarity Matrix

	Bob	Ted	Ann
Bob	1	0	0.54
Ted	0	1	0.55
Ann	0.54	0.55	1

The 2D matrix depicted above shows the cosine similarity scores computed between all pairs of users within our system.

After computing the cosine similarity scores between all users and arranging them in descending order, we can create a list sorted by these similarity scores. By selecting the top N neighbors from this list, we effectively find the users who exhibit the highest similarity to a given user. For instance, in the example, Ann emerges at the top of the list for Bob due to their significant similarity. However, we might opt to discard Ted from consideration by imposing a minimum similarity score threshold to ensure that only the most similar users are considered as neighbors. This step helps in creating a reliable and relevant set of user-based recommendations.
Here's a quick recap of the steps involved in user-based collaborative filtering:

Recap

- **Job Seeker ⇒ Job Offer Rating Matrix**
- **Job Seeker ⇒ Job Seeker Similarity Matrix**
- **Find Similar Job Seekers**
- **Candidate Generation**
- **Candidate Scoring**
- **Candidate Filtering**

KNN Recommendations

The user-based K-nearest neighbors (KNN) approach for predicting job seeker ratings to a specific job offer involves the following steps.

Given a user-item matrix R, where R_{ui} represents the rating given by user u to item i. Let $N(u)$ be the set of users similar to user u based on some similarity measure, and let k be the number of nearest neighbors to consider for prediction.

The predicted rating $\hat{R}_{ui}$ for user u and item i can be calculated using the following steps:

1. Calculate the similarity score between users: For each pair of users u and v, compute their similarity score $s(u, v)$ using a similarity metric such as the Pearson correlation coefficient or the cosine similarity.

2. Select the nearest neighbors: Sort the users in descending order based on their similarity scores with user u, and select the top k users as the nearest neighbors, denoted as $N_k(u)$.

3. Calculate the weighted average: Compute the predicted rating $\hat{R}_{ui}$ for item i by taking the weighted average of the ratings of the nearest neighbors for that item:

$$\hat{R}_{ui} = \frac{\sum_{v \in N_k(u)} s(u, v) \cdot R_{vi}}{\sum_{v \in N_k(u)} |s(u, v)|}$$

4. Normalize the predicted rating: If necessary, normalize the predicted rating $\hat{R}_{ui}$ to the appropriate rating scale (e.g., if the original ratings are integers from 1 to 5, round $\hat{R}_{ui}$ to the nearest integer).

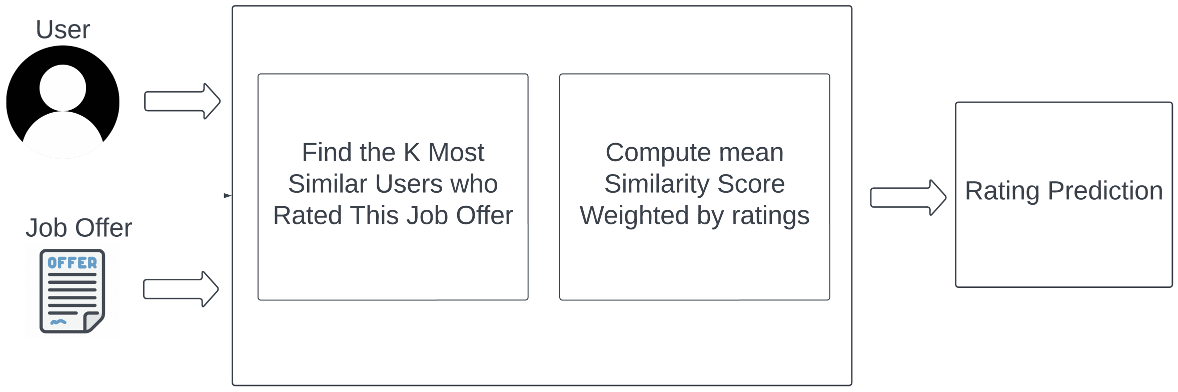

User
Job Offer
OFFER
Find the K Most Similar Users who Rated This Job Offer
Compute mean Similarity Score Weighted by ratings
Rating Prediction

Figure 4.9: KNN User Based Recommendation

Interpretation

By leveraging the KNN algorithm, we gain valuable insights into predicting job seeker preferences, which ultimately aids in creating tailored and accurate job recommendations for individual users.

Item-Based Collaborative Filtering

Item-based collaborative filtering is an alternative approach to collaborative filtering that flips the problem on its head. Instead of finding similar users to make recommendations, it focuses on the Job offers that a user liked and recommends other offers that are similar to those. Hence, it is called item-based collaborative filtering, in contrast to the user-based approach.

There are several reasons why utilizing similarities between items can be advantageous over similarities between users.

Firstly, job offers tend to maintain a more stable and consistent nature compared to individuals. A job offer for a data science role will always pertain to data science, while a person's preferences may fluctuate and change rapidly throughout their life.

Additionally, constructing item similarities provides a significant advantage in terms of data volume. Typically, there are far fewer job offers to consider in comparison to the vast number of job seekers. As a result, working with item similarities reduces the computational complexity involved in the recommendation process, making it more efficient and manageable.

Also, the user experience for new visitors to a website is significantly enhanced in item-based filtering. As soon as a new user shows interest in a particular item, the system can promptly recommend other items that are similar to the one they liked. In contrast, if user-based collaborative filtering were utilized, there would be no recommendations available for a new user until they are included in the subsequent update of the user similarity matrix.

TopN Recommendations

In item-based collaborative filtering, the general process remains similar to the user-based collaborative filtering, but the key difference lies in how we use users and job offers. Instead of finding similar users to a target user and recommending offers they liked, we focus on the job offer the target user liked and recommend other similar offers.

Table 4.3: Offer-User Rating Matrix

	Bob	Ted	Ann
Data Engineer Offer	5	0	4
Data Analyst Offer	0	5	5
Data Science Offer	0	5	0
Cloud Computing Offer	0	4	0

we rearrange the matrix representation, making items the rows and users the columns. This rearrangement allows us to efficiently access all the user ratings for a specific item, enabling us to measure the similarity between items based on the users who rated them. By doing so, we can identify patterns and connections among items that are preferred by similar groups of users, enhancing the accuracy of our recommendations.

Table 4.4: Offer Similarity Matrix

	Data Engineer	Data Analyst	Data Science	Cloud Computing
Data Engineer	1	0.44	0	0.7
Data Analyst	0.44	1	0.7	0
Data Science	0	0.7	1	0
Cloud Computing	0	0	0	1

conceivable pair of job offers. By exploring these similarity scores, we can pinpoint job offers that are favored by similar groups of users and make more precise and relevant recommendations.

Imagine our user, Bob, who has just indicated an interest in a data engineer job offer. While we might not have much information about Bob yet, we can make use of our item-to-item similarity matrix to find other job offers that are similar to the data engineer offer. This is achieved by considering the ratings of other job seekers who have shown interest in both the data engineer and other job offers in the past. Surprisingly, this approach leads us to discover the data analyst offer, which exhibits similarities in user preferences. By leveraging item-based collaborative filtering, we can swiftly offer Bob additional job suggestions that align with his initial interest.

Item-Based KNN

Item-based KNN operates similarly like the User-Based approach, but we interchange ”job seekers” and ”job offers” in the process. To forecast the rating of a user, let’s say Bob, for a particular job offer, say a Data Engineer position, we begin with a collection of k items that Bob has also rated, exhibiting the highest similarity to the Data Engineer job offer.

The following steps mirror the user-based approach. We calculate the weighted average of the similarity scores based on the ratings and obtain a sound rating prediction. The item-based KNN method proves to be effective in providing job seekers, like Bob, with relevant job recommendations based on their preferences and past interactions with job offers.

1. Calculate the similarity score between job offers: For each pair of job offers i and j, compute their similarity score $s(i, j)$ using a similarity metric such as the Pearson correlation coefficient or the cosine similarity.
2. Select the most similar job offers: Sort the offers in descending order based on their similarity scores with offer i, and select the top k offers as the most similar offers, denoted as $N_k(i)$.
3. Predict the user’s rating for offer i: Compute the predicted rating $\hat{R}_{ui}$ for job offer i by taking the weighted average of the ratings of the user u for the most similar job offers $N_k(i)$:

$$\hat{R}_{ui} = \frac{\sum_{j \in N_k(i)} s(i,j) \cdot R_{uj}}{\sum_{j \in N_k(i)} |s(i,j)|}$$

4. Normalize the predicted rating: If necessary, normalize the predicted rating $\hat{R}_{ui}$ to the appropriate rating scale (e.g., if the original ratings are integers from 1 to 5, round $\hat{R}_{ui}$ to the nearest integer).

4.1.5 Matrix Factorization for Recommendation

Within the realm of matrix factorization techniques, numerous methods exist to extract valuable insights from data. However, for our recommender system, we have chosen to employ the most widely acclaimed and favored approach—the Singular Value Decomposition (SVD) algorithm.

Principal Component Analysis

Principal Component Analysis[3] (PCA) is a powerful technique often referred to as a ”dimensionality reduction” approach. The primary objective of PCA is to transform high-dimensional data, such as the multitude of job offers a user might rate, into a lower-dimensional representation that effectively captures the essential characteristics of a job offer. By doing so, we aim to retain the most important information while reducing the computational burden and potential noise associated with the original high-dimensional data.

The concept of ”principal components” lies at the core of PCA. These components are linear combinations of the original features (such as required experience, salary, or job role) that capture the maximum variance in the data. Each principal component represents a different direction in the data space, and together, they form an orthogonal basis that defines a new coordinate system.

SVD

The Singular Value Decomposition[7] (SVD) is a fundamental matrix factorization technique used to decompose a given matrix into three constituent matrices. It is widely employed in various fields, including data analysis, machine learning, and recommender systems.

In simple terms, given a matrix A, the Singular Value Decomposition (SVD) decomposes it into three matrices: U, Σ, and V^T (transpose of V). Here’s the breakdown of each matrix:

- U: The matrix U contains the left singular vectors. These vectors represent the column space of the original matrix A. Each vector in U represents how the data points are distributed along the orthogonal directions in the row space of A.
- Σ: The diagonal matrix Σ contains the singular values of the original matrix A. The singular values represent the scaling factors along the principal axes of the data, and

they are ordered from the largest to the smallest. The singular values provide crucial information about the importance of each dimension in the data.

- V^T: The matrix V^T (transpose of V) contains the right singular vectors. Similar to U, these vectors represent the row space of the original matrix A. Each vector in V^T represents how the data points are distributed along the orthogonal directions in the column space of A.

How It Works?

Imagine we have a large matrix where the rows represent job seekers, the columns represent job offers, and the cells contain ratings given by job seekers to the corresponding job offers. However, this matrix is likely to be very sparse because each job seeker has only rated a few job offers, leaving many cells empty.

Table 4.5: User-Job Offer Rating Matrix

	Data Engineer	Data Analyst	Data Science	Cloud Computing
Bob	5	0	0	0
Ted	0	5	5	4
Ann	4	5	0	0

Subsequently, we perform PCA (Principal Component Analysis) on our multidimensional job offers rating dataset matrix R, where each dimension represents a unique job offer. We refer to this transformed dataset as the ”ratings matrix,” with users being the rows and the principal components representing the job offers. We shall denote the resulting matrix from PCA as U.

Similar to how we can apply PCA on our user ratings data to identify typical user profiles, we can also flip the approach and use PCA to discover typical profiles of job offers.

Table 4.6: Job Offers and Ratings

	Bob	Ted	Ann
Data Engineer Offer	5	0	4
Data Analyst Offer	0	5	5
Data Science Offer	0	5	0
Cloud Computing Offer	0	4	0

Let’s call this matrix above R^T. After applying PCA on the transpose of the rating matrix, we obtain a resulting matrix, which we’ll call M. This matrix provides a compact and dense representation of job offers through vectors. Each vector in matrix M captures essential information about a job offer, effectively describing its characteristics in a reduced-dimensional space. This dimensionality reduction enables us to efficiently work with the data and identify patterns or similarities among job offers, simplifying the recommendation process.

Now, after measuring the matrices M and U, the SVD enables us to represent the original matrix (job seeker-job offer interactions) and obtain the predicted rating for each offer by each user as a product of these matrices. Let's call this matrix R (Rating Matrix):

$$R \approx M \Sigma U^T$$

In our current setup, each rating in the original ratings matrix R can be expressed as the dot product of a row from the matrix M and a column from the transposed matrix U^T. To make accurate predictions, we need to infer the values for the complete rows and columns in M and U^T based on the known ratings in R.

This process can be approached as a minimization problem, where we aim to find the optimal values for the missing elements in M and U^T that minimize the errors between the known ratings in R and our predictions. This problem falls within the realm of machine learning, and there are various techniques available to address it effectively.

One commonly used method for solving this problem is Stochastic Gradient Descent (SGD). SGD iteratively adjusts the parameters of the matrices M and U^T to gradually minimize the prediction errors. By updating these parameters in the direction that leads to reduced errors, SGD efficiently converges to a solution that closely matches the observed ratings while inferring the missing ones.

4.1.6 Deep Learning For Recommendation

Indeed, deep learning offers a powerful approach for recommendation systems, and in this chapter, we will explore its application in depth. Leveraging the advantages of neural networks, we will delve into the intricacies of using deep learning to enhance recommendation techniques. By harnessing the potential of neural networks, we aim to achieve more accurate and sophisticated recommendations that cater to the unique preferences and interests of users.

Restricted Boltzmann Machine

Restricted Boltzmann Machines (RBMs)[15] are among the simplest neural networks, comprising just two layers: a visible layer and a hidden layer. During training, the training data is fed into the visible layer in a forward pass, and weights and biases between the layers are adjusted during backpropagation. An activation function like ReLU is used to generate the output from each hidden neuron.

The name ”Restricted Boltzmann Machines” stems from the fact that neurons within the same layer cannot communicate directly, limiting connections only between the two distinct layers. This restriction aligns with modern neural networks' practices, but it wasn't present in earlier Boltzmann Machines during the early stages of AI development. Additionally, the name Boltzmann references the Boltzmann distribution function used for their sampling function. The credit for RBMs goes to Geoffrey Hinton, a professor at Carnegie Mellon University, with the concept dating back to 1985.

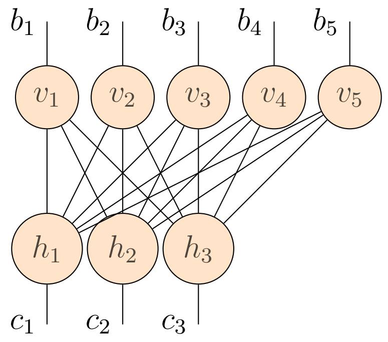


Figure 4.10: Restricted Boltzmann Machine Architecture

How it Works?

In the context of recommendation systems, each Restricted Boltzmann Machine (RBM) represents a single user and takes as input all the available job offers. During the training process, RBMs are specifically designed to deal with missing data, which can occur when certain job offers are not seen by the user.
Through an iterative learning process, RBMs adjust the connections between visible and hidden nodes in order to learn patterns and dependencies in the data. By doing so, they are able to reconstruct the input data, effectively predicting the job offers a user may have missed or not interacted with. Once the RBM is trained, it can utilize its learned representations to generate new ratings by sampling from its hidden layer, thereby providing personalized job recommendations based on the user's preferences and past interactions with items. This ability to handle missing or incomplete information, coupled with their capacity to capture intricate data dependencies, makes RBMs powerful tools for creating accurate and effective recommendation systems. By leveraging RBMs, recommendation engines can deliver more tailored and relevant suggestions, enhancing the overall user experience and satisfaction.

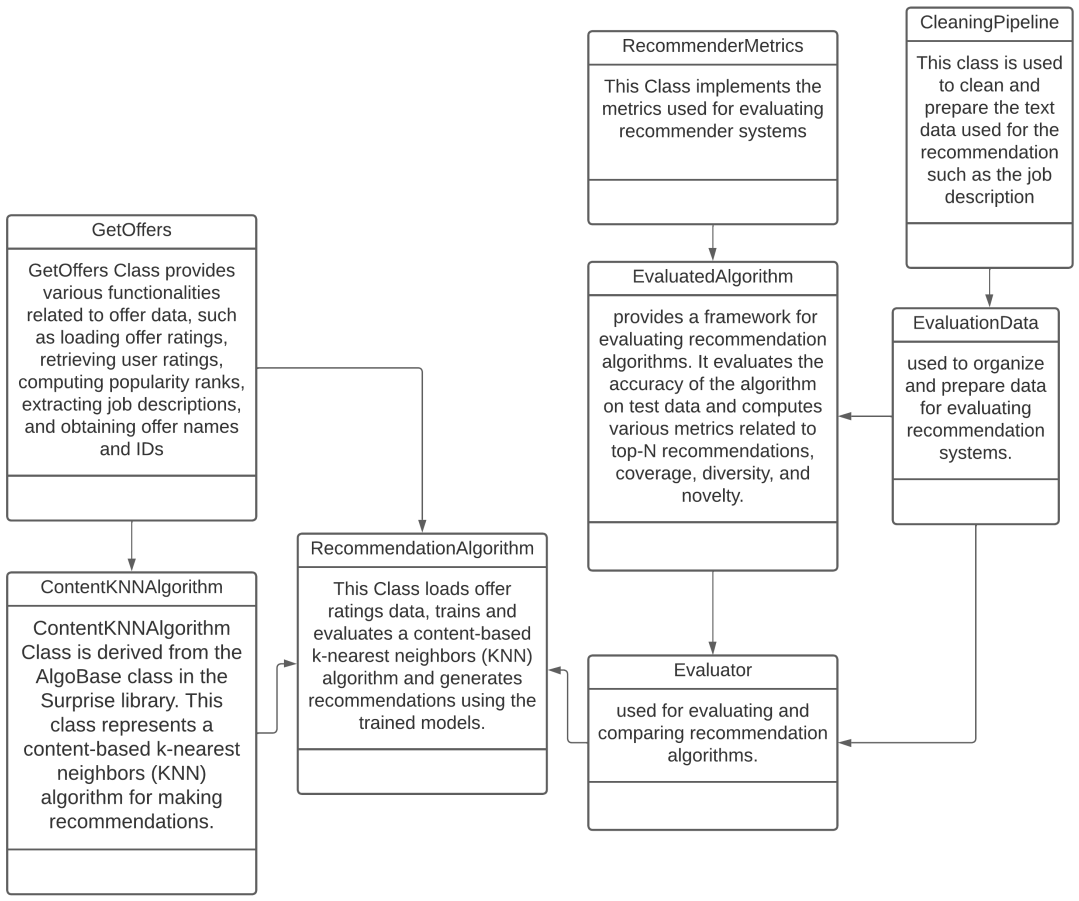
4.2 Implementation and Execution

In order to build a robust and efficient recommendation system, we require a versatile framework that enables seamless experimentation, evaluation, and comparison of different algorithms. SurpriseLib emerges as the ideal choice, as it caters precisely to these needs. It equips us with the necessary tools and flexibility to implement, test, and refine various recommender system models. By building upon this foundation, we can unlock the full potential of our recommendation engine and deliver superior user experiences.

Also, we embraced the Object-Oriented Programming (OOP) paradigm to facilitate modularity, maintainability, and extensibility. By employing OOP principles, we structured the system into classes that encapsulate distinct functionalities, enabling seamless integration and collaboration among components. In the following of section, we explore the design of our recommender system, encompassing both the KNN and TopN architectures.

4.2.1 KNN-Based Recommender Engine Design

Let's commence with the KNN Recommender Engine design, depicted in the following class diagram.



RecommenderMetrics
This Class implements the metrics used for evaluating recommender systems
CleaningPipeline
This class is used to clean and prepare the text data used for the recommendation such as the job description
GetOffers
GetOffers Class provides various functionalities related to offer data, such as loading offer ratings, retrieving user ratings, computing popularity ranks, extracting job descriptions, and obtaining offer names and IDs
EvaluatedAlgorithm
provides a framework for evaluating recommendation algorithms. It evaluates the accuracy of the algorithm on test data and computes various metrics related to top-N recommendations, coverage, diversity, and novelty.
EvaluationData
used to organize and prepare data for evaluating recommendation systems.
ContentKNNAlgorithm
ContentKNNAlgorithm Class is derived from the AlgoBase class in the Surprise library. This class represents a content-based k-nearest neighbors (KNN) algorithm for making recommendations.
RecommendationAlgorithm
This Class loads offer ratings data, trains and evaluates a content-based k-nearest neighbors (KNN) algorithm and generates recommendations using the trained models.
Evaluator
used for evaluating and comparing recommendation algorithms.

Figure 4.11: KNN Recommendation Algorithm Design

The Class Diagram of our system illustrates the key classes and their relationships. At the core, we have the GetRecommendation class, serving as the central orchestrator for recommendation tasks. This class interacts with various other classes, including KNNAlgorithm, Evaluator, and GetOffers.

Notice

Due to the large number of methods in my class, I have chosen to represent only the functionality of each class in the class diagram, rather than including all its methods and attributes.

⇒ **By structuring our recommender system with these classes, we achieve a modular and organized architecture that promotes code reusability, ease of testing, and the ability to add new algorithms seamlessly. Each class handles specific functionalities, making the system more maintainable and extensible as we continue to explore and refine different recommendation techniques.**

4.2.2 TopN-Based Recommender Engine Design

In the TopN architecture, the overall design of the recommender system remains similar to the KNN architecture, with a few key modifications. The classes and modules responsible for data handling, user-item interactions, and recommendation tasks are still present, providing continuity and consistency between the two approaches.

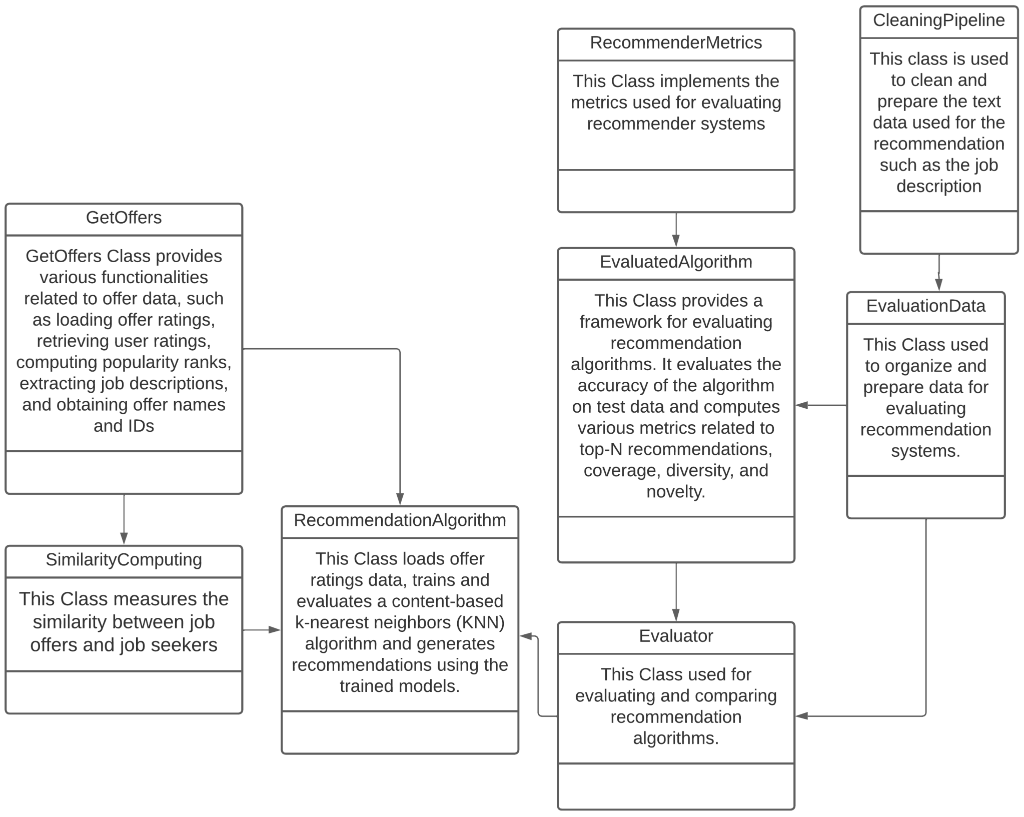

RecommenderMetrics
This Class implements the metrics used for evaluating recommender systems
CleaningPipeline
This class is used to clean and prepare the text data used for the recommendation such as the job description
GetOffers
GetOffers Class provides various functionalities related to offer data, such as loading offer ratings, retrieving user ratings, computing popularity ranks, extracting job descriptions, and obtaining offer names and IDs
EvaluatedAlgorithm
This Class provides a framework for evaluating recommendation algorithms. It evaluates the accuracy of the algorithm on test data and computes various metrics related to top-N recommendations, coverage, diversity, and novelty.
EvaluationData
This Class used to organize and prepare data for evaluating recommendation systems.
RecommendationAlgorithm
This Class loads offer ratings data, trains and evaluates a content-based k-nearest neighbors (KNN) algorithm and generates recommendations using the trained models.
SimilarityComputing
This Class measures the similarity between job offers and job seekers
Evaluator
This Class used for evaluating and comparing recommendation algorithms.

Figure 4.12: TopN Recommender Engine Design

The crucial difference in the TopN architecture lies in the recommendation algorithm. In the KNN architecture, we utilized the KNNAlgorithm module to perform user-based or item-based collaborative filtering using similarity calculations. However, in the TopN architecture, we introduce the SimilarityComputing class to handle the calculation of similarity scores between items or job offers.

The SimilarityComputing class serves as the core engine for recommendation in the TopN architecture. It uses various techniques, such as cosine similarity or other appropriate similarity metrics, to measure the similarity between the attributes of job offers and candidates.

Review

The two architectures, KNN-based and TopN, serve as the foundation for both content-based filtering and collaborative filtering in our recommender system. They enable us to leverage user preferences and item similarities to provide personalized job recommendations to job seekers.

4.3 Results and Finding

In this section, we present the outcomes of our recommender system results. A comprehensive analysis of the performance and effectiveness of our recommendation models is provided, offering insights into how well they align with the objectives of our study.

In the testing phase, we conducted recommendation experiments for a Job seeker with a data scientist profile. Our recommender system utilized the dataset of all collected job offers to identify and recommend relevant job opportunities that align with the data scientist's expertise and preferences.

4.3.1 KNN Recommendation Results

In the table below, you can find the results of the content-based collaborative filtering using the KNN architecture. The table displays the recommended jobs along with their corresponding predicted ratings

Table 4.7: KNN Content-Based Filtering Results

Recommendations	Predicted Rating
Data Scientist	4.425832175373266
Sr. Data Scientist	4.424483444081695
Data Scientist	4.4197811646774925
Product Data Analyst	4.419094068362349
Lead Data Scientist, R&D Team	4.417387597605685
Data Analyst	4.4170997150017115
Data Scientist	4.415798709508128
Data Scientist	4.415156619069042
Data Scientist	4.414271478218878
VP, Data Scientist	4.413804828608584

In these next two tables, the results of collaborative filtering using the KNN algorithm are presented. The system has employed user-based and Item Based collaborative filtering to recommend job offers to a specific user, considering their historical ratings and preferences. Each recommended job offer is accompanied by a predicted rating, which indicates how well it is expected to align with the user's interests. The table displays the job offer titles and their respective predicted ratings, allowing the user to explore personalized recommendations tailored to their preferences.

Table 4.8: KNN User-Based Collaborative Filtering Results

Recommendations	Predicted Rating
Data Scientist	4.435074146776219
Statistician (Data Scientist)	4.395046612787354
Data Scientist	4.395028473238585
Data Scientist	4.365233219894231
Machine Learning Engineer, Computer Vision	4.364938648980923
Senior Data Scientist	4.357558299188082
Data Scientist, Buyer Experience	4.349985865580645
Quantitative Researcher - Alpha Capture	4.347588920158549
Data Scientist, Marketplace Economics	4.34008179034289
Data Scientist, Voter Data	4.337535550506799

Table 4.9: KNN Item-Based Collaborative Filtering Results

Recommendations	Predicted Rating
Data Science Specialist USA	4.4758009959789411
Sr Data Scientist	4.45745654094831
Machine Learning Engineer, Computer Vision	4.454892267562965
Scientist Data Analytics	4.4524724577966515
Statistician (Data Scientist)	4.450002700153933
Data Scientist	4.444897318638816
Data Scientist, Recommendations	4.442519782543689
Data Scientist	4.439937930280396
Data Analyst	4.43256956932881

In the following two tables, we present the results of matrix factorization using the Singular Value Decomposition (SVD) algorithm. On the left side, the system employed the conventional SVD approach, while on the right side, it used the enhanced SVD++[8] algorithm. Both methods have been used to find latent factors that capture the relationships between job offers and user preferences, resulting in accurate predictions of job offer ratings. Users can explore personalized recommendations generated by each approach, facilitating better decision-making in their job search.

Table 4.10: SVD Results

Recommendations	Predicted Rating
Data Scientist	4.594390523306852
Data Scientist (Java)	4.561862971663752
Data Science Analyst	4.552484436598275
Data Scientist/Machine Learning	4.548846064382402
Software Engineer - Data Engineering, Finance Products	4.530252563025661
AI Engineering: Data Scientist/Machine Learning Engineer	4.527474259957528
Data Scientist, Decisions	4.509825716886096
Quantitative Researcher	4.505127878894924
Data Scientist/Data Science Instructor (Part-time)	4.493747829744327
Data Scientist/Data Architect	4.485221899375255

Table 4.11: SVD++ Results

Recommendations	Predicted Rating
Data Scientist	4.548089059224887
Technical Data Consultant	4.485303410377958
Data Scientist, Marketplace Economics	4.475899130212679
Data Engineer	4.472740417044908
Data Scientist, Acorn AI Labs	4.4677359860451835
Senior Data Scientist, Real World EvidenceNY	4.464480172492173
Data Scientist, Voter Data	4.462619662837714
Data Scientist, Decisions	4.4624205682078815
AI Engineering: Data Scientist/Machine Learning Engineer	4.456574612260545
Quantitative Researcher	4.4553675164905435

In the table under, the results of recommendations using the Restricted Boltzmann Machine (RBM) algorithm are displayed. RBMs have been used to model the relationships between job offers and user preferences. Each recommended job offer is accompanied by a predicted rating, generated by the RBM model based on the user's interactions with the dataset. The table presents the job offer titles and their respective predicted ratings, enabling the user to explore personalized and accurate recommendations tailored to their profile.

Table 4.12: Restricted Boltzmann Machine Results

Recommendations	Predicted Rating
Data Scientist/Lead Data Scientist	3.2789946
Technical Data Consultant	3.252274
Data Scientist - Analytics Engineer	3.2479477
Data Scientist - Forecasting	3.238458
Principal Data Scientist	3.2181091
Data Scientist, Acorn AI Labs	3.212473
Data Scientist	3.2110229
Data Scientist	3.209681
Data Scientist	3.2056859
Data Analyst	3.2046833

4.3.2 TopN Recommendation Results

Table 4.13: TopN User-Based Collaborative Filtering Results

Recommended Jobs	Similarity Score
Quantitative Researcher - Alpha Capture	51.66
Principal Data Scientist	47.66
Lead Data Scientist, R&D Team	47.24
Data Scientist, Recommendations	47.01
Machine Learning Engineer, Computer Vision	47.01
Data Scientist/Lead Data Scientist	46.87
Data Scientist	46.07
Data Scientist	45.09
Data Scientist	44.95
Technical Data Consultant	44.83
Data Scientist	44.68

In the Table above, the results of the user-based collaborative filtering using the TopN architecture are displayed. The system has recommended job offers for a specific user, who is a data scientist. The recommendations are based on the similarity of this user's preferences and behavior to other users in the dataset. The table shows the titles of the recommended job offers, along with their corresponding similarity scores, indicating how closely they align with the user's interests. The user can explore these recommendations to find job offers that match their profile and preferences.

Table 4.14: TopN Item-Based Collaborative Filtering Results

Recommended Jobs	Similarity Score
Data Scientist, Analytics	61.54
Data Scientist	61.52
AI Engineering: Data Scientist/Machine Learning Engineer	61.51
Quantum Data Scientist (Industrial-Process sector)	61.50
Data Analyst	61.50
Lead Data Scientist	61.49
Principal Data Scientist	61.49
Sports Betting Data Scientist	61.49
Data Analyst	61.48
Quantitative Researcher - Alpha Capture	61.48
Data Scientist	61.48

In this Table, the results of the item-based collaborative filtering using the TopN architecture are presented. The system has recommended job offers that are similar to the job offer that the user liked. The recommendations are based on the similarity of job offers' attributes, such as required experience and job description. The table displays the titles of the recommended job offers and their respective similarity scores, indicating how closely they match the attributes of the selected job offer. Job seekers can use these recommendations to discover similar job opportunities that align with their skills and interests.

4.4 Lesson Learned and Reflection

Throughout this chapter on recommender systems, I have embarked on a rewarding journey, unravelling the complexities involved in crafting efficient recommendation models. The significance of algorithm selection has been underscored, guiding me to tailor the choice to match specific use cases while maintaining a balance between functionality and project complexity. Dealing with data scarcity has been a crucial challenge, pushing me to employ creative solutions such as sparse arrays to handle missing values and ensure reliable recommendations. Emphasizing robustness, I learned the importance of adhering to software engineering best practices, resulting in a well-structured and maintainable recommender engine. This foresight will not only enhance the system's performance but also facilitate future upgrades and adaptations as the project evolves. The journey has ignited my passion for data science, reinforcing the art of designing user-centric systems that cater to diverse preferences and enable seamless user experiences.

Conclusion and Next Step

After successfully creating a variety of recommender systems that cater to different needs and scenarios, the next crucial phase is to evaluate these systems meticulously in the Evaluation chapter, by conducting rigorous assessments of their performance, accuracy, and ability to achieve predefined objectives, we can gain valuable insights into their effectiveness. In parallel, in the next chapter, I will be focusing on the development of an AI-based solution tailored to assist recruiters in selecting the most qualified candidates for their job offers. This innovative system will leverage advanced algorithms and techniques to analyze candidate profiles, job requirements, and other relevant data. By employing machine learning and data-driven insights, the solution aims to streamline the candidate selection process, providing recruiters with valuable assistance and empowering them to make informed decisions. Through the integration of cutting-edge technologies, this AI-driven solution aspires to revolutionize the recruitment landscape, making it more efficient, accurate, and successful for both employers and job seekers alike.

Chapter 5

Candidates Selection

Introduction

In this chapter, we thoroughly explore the intricacies and workings of our candidates' selection and ranking system. We focus on creating a robust and efficient solution to streamline the recruitment process for job offers.

5.1 Banckground and Context

Our Candidates' ranking system is designed to take a job offer as input, encompassing essential attributes like required years of experience, salary, and a comprehensive job description. Additionally, the system receives multiple candidate applications, along with their resumes and other pertinent attributes. Leveraging advanced algorithms, the system efficiently matches and ranks the Top N candidates who best align with the specific requirements of the job offer. By analyzing and comparing the job attributes and candidate profiles, this solution enhances the recruitment process, ensuring that recruiters can quickly identify the most qualified and suitable candidates, thereby saving valuable time and effort in the hiring process.

The figure below showcases the core of the system:

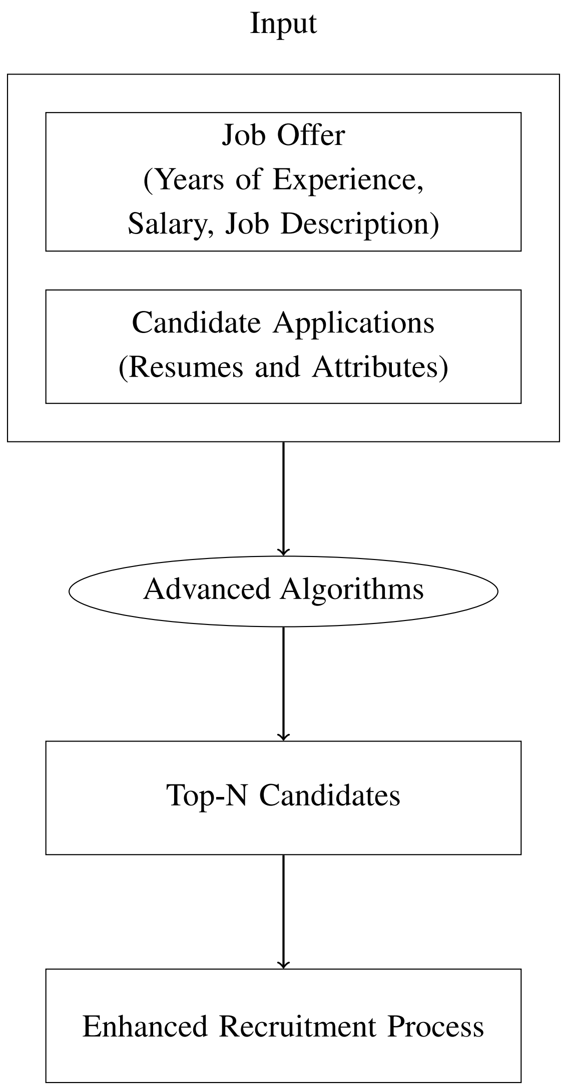

Input
Job Offer (Years of Experience, Salary, Job Description)
Candidate Applications (Resumes and Attributes)
Advanced Algorithms
Top-N Candidates
Enhanced Recruitment Process

Figure 5.1: Illustration of the candidates' selection System

Now let's delve deep into the core of our candidate matching algorithm to understand how the AI system selects the top N candidates. The algorithm leverages machine learning techniques and data analysis to evaluate each candidate's qualifications in relation to the job offer's requirements.

How It Works?

The candidate selection process of the system commences by taking input, including the applicants and the job offer details.

First of all, through a text cleaning pipeline introduced in Chapter 3, the job description and candidates' resumes undergo preprocessing to eliminate irrelevant information and noise. Subsequently, the text data is transformed using the trained word2vec algorithm, as discussed in the same chapter, yielding vectors that represent each candidate's resume and the job offer description. These vectors share the same length and encode the semantic meaning of each resume.

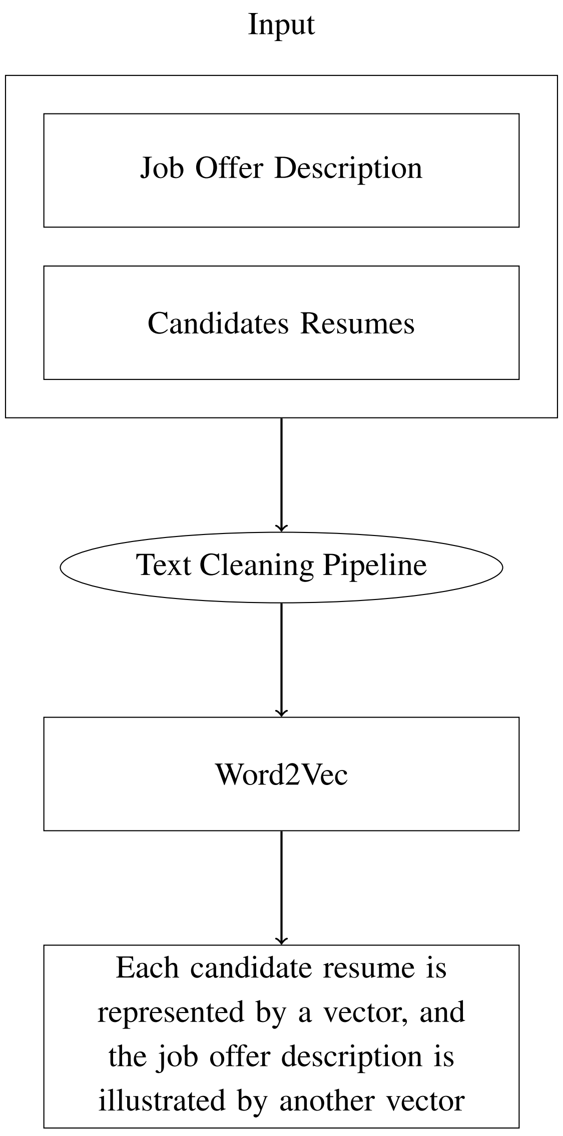

Input
Job Offer Description
Candidates Resumes
Text Cleaning Pipeline
Word2Vec
Each candidate resume is represented by a vector, and the job offer description is illustrated by another vector

Figure 5.2: Illustration of the Text Preparation Steps

Using cosine similarity, the system calculates the similarity between each candidate's resume vector and the job description vector. This similarity score reflects the degree to which a candidate's resume aligns with the specified job description. Simultaneously, the system

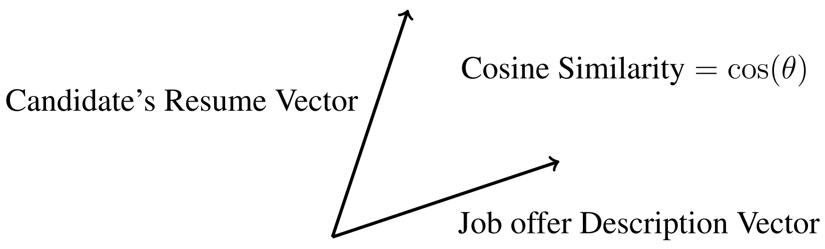

Cosine Similarity $= \cos(\theta)$
Candidate's Resume Vector
Job offer Description Vector

Figure 5.3: Candidates' Selection Cosine Similarity Illustration

computes the similarity between each candidate's years of experience and the required job experience, producing a similarity value between 0 and 1. The equation to-do this task is given by:

$$f(x_1, x_2) = \exp\left(\frac{|x_1 - x_2|}{3}\right)$$

In this equation, x_1 represents the job offer's required experience, while x_2 represents the candidate's number of years of experience. The function $f(x_1, x_2)$ gives a measure of compatibility, with higher values indicating a closer match between the job requirements and the candidate's experience.

Notice

To obtain a comprehensive candidate assessment, the system multiplies the text similarity scores, representing the match between candidates' resumes and the job description, with the similarity scores based on years of experience. This combination accounts for both textual qualifications and experience relevance.

The system then ranks the candidates based on their combined similarity scores and presents the top N candidates that best match the job offer's requirements. By leveraging advanced text analysis and semantic understanding, the system efficiently identifies the most suitable candidates, significantly streamlining the candidate selection process for recruiters. The integration of text-based similarity and experience relevance ensures that the selected candidates closely align with the specific job offer, optimizing the likelihood of successful job placements.

5.2 Implementation and Execution

In the implementation and execution of the resume ranking service, I adopted a modular programming approach to ensure a well-organized and maintainable system. By breaking down the entire process into smaller, manageable modules, I aimed to enhance code reusability, readability, and overall system efficiency. The system design revolves around the seamless integration of these modules, each responsible for specific tasks, creating a cohesive and robust workflow.

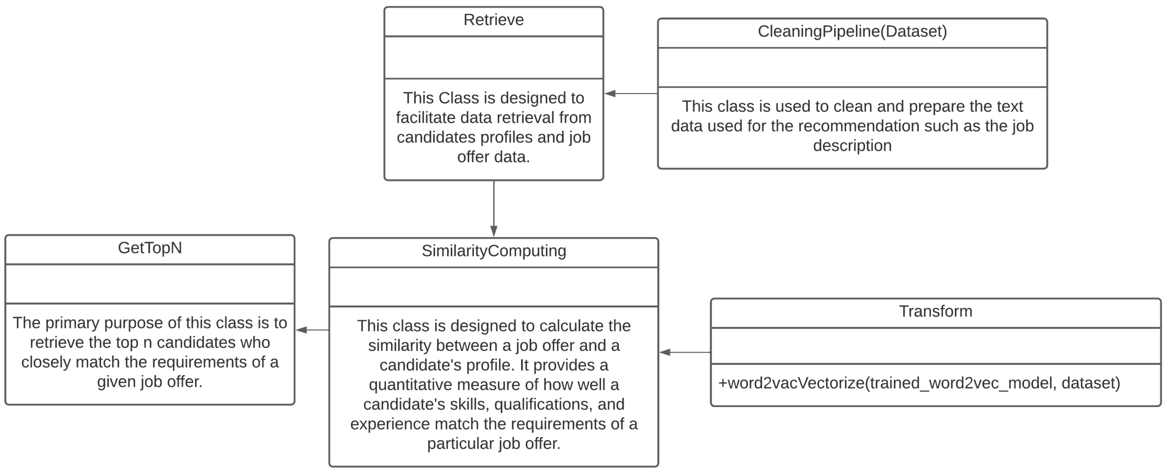

Retrieve
This Class is designed to facilitate data retrieval from candidates profiles and job offer data.
CleaningPipeline(Dataset)
This class is used to clean and prepare the text data used for the recommendation such as the job description
GetTopN
The primary purpose of this class is to retrieve the top n candidates who closely match the requirements of a given job offer.
SimilarityComputing
This class is designed to calculate the similarity between a job offer and a candidate's profile. It provides a quantitative measure of how well a candidate's skills, qualifications, and experience match the requirements of a particular job offer.
Transform
+word2vacVectorize(trained_word2vec_model, dataset)

Figure 5.4: Candidates' Selection System Design

The first module CleaningPipeline focuses on text preprocessing, where I leveraged the text cleaning pipeline, we discussed in the earlier chapters to ensure consistent and standardized input. This module handles tasks such as removing stop words, stemming, and tokenization, effectively preparing both the job description and candidates' resumes for further analysis.

The next crucial module is called ”Retrieve,” which plays a pivotal role in data handling and preparation. This module is responsible for loading the raw data into a pandas dataframe, creating a structured and organized data representation. Once the data is in the dataframe, the ”Retrieve” module calls upon the ”CleaningPipeline” module to apply the text cleaning pipeline we designed earlier. The next module involved word2vec transformation, utilizing the trained word embeddings model from Chapter 3. This step allowed me to convert the textual data into dense vectors that capture semantic meanings. With vectors representing both the job description and candidates' resumes, we now have a foundation for similarity computation.

The core of the resume ranking service lies in the SimilarityComputation module by implementing the ”Retrieve” module, which ensures that the data is loaded and preprocessed into a clean dataframe. Additionally, the ”SimilarityComputing” class takes charge of the ”Transform” module, which involves vectorizing the cleaned dataframe. This vectorization process enables us to calculate text similarity scores between the job description and the candidates' resumes. Moreover, this class also handles the calculation of similarity based on the years of experience of the candidates, providing a comprehensive evaluation of the candidates' qualifications.

Finally, the ranking aggregation module named GetTopN combines text similarity and experience relevance scores to generate a holistic ranking for each candidate. The system then presents the top N candidates that best match the job offer's requirements, streamlining the candidate selection process for recruiters.

To enhance the flexibility and user-friendliness of our resume ranking service, we have implemented a YAML configuration file that empowers the recruiter to customize the number of candidates they want to retrieve, denoted as N. This configuration file acts as a simple and intuitive interface for the recruiter to fine-tune the candidate selection process without altering the codebase. By introducing this YAML-based configuration mechanism, we empower recruiters to adjust N effortlessly without the need for any technical expertise or code modification.

Remark

By implementing a modular design, I not only achieved a well-structured and organized system but also set the foundation for future expansions and enhancements. This approach fosters code maintainability and ease of integration with other components, providing a scalable and adaptable resume-ranking service.

5.3 Results and Finding

In the Results and Findings section of our resume ranking system, we showcase the practical outcomes yielded by our AI-driven solution. Through real-world scenarios involving a 'data science' job offer and a dataset of candidates who applied for it, we set the candidate selection limit to 5 (N=5). The system accurately ranks and presents the top 5 candidates based on their resumes' semantic similarity to the job description and the experience similarity.

Table 5.1: Candidates' Selection Results

Applicant Id	Similarity Score	Applicant Name
7	0.137113	Nicholas Daugherty
38	0.126532	Amanda Brown
18	0.126532	Mark Garrett
21	0.114760	Michele Christian
1	0.069606	Steven Glenn

The provided results display the full names and IDs of the selected candidates, allowing recruiters to access their detailed information promptly. Moreover, the system presents the similarity scores for each candidate, indicating the degree of alignment between their qualifications and the job requirements.

These case studies demonstrate the system's effectiveness in streamlining the candidate selection process, significantly saving recruiters' valuable time and effort. By leveraging our resume ranking system, recruiters can swiftly identify the most qualified applicants, ensuring a more efficient and informed hiring decision-making process. The reliable and data-driven results highlight the system's potential to revolutionize candidate selection practices and enhance the overall recruitment experience.

5.4 Lesson Learned and Reflection

Through this chapter, we gained invaluable insights into the development and implementation of our resume ranking system. Firstly, we recognized the significance of employing modular programming, which facilitated the system's design and ensured flexibility for future upgrades and modifications. Emphasizing code modularity allowed us to efficiently manage different modules responsible for data retrieval, text cleaning, similarity computation, and ranking.

Additionally, we acknowledged the importance of leveraging advanced Natural Language Processing techniques such as word2vec and cosine similarity to transform and measure the semantic similarity between job descriptions and candidates' resumes. These techniques significantly enhanced the accuracy and efficiency of our candidate selection process. Furthermore, providing flexibility for recruiters by enabling them to adjust the number of candidate selections (N) through a YAML configuration file proved to be a practical and user-friendly approach. This adaptable feature empowers recruiters to customize the system to their specific needs without altering the core codebase.

Notice

Evaluating the resume ranking services is a complex task because it is inherently subjective and context-dependent, as different recruiters have diverse preferences and specific criteria for candidate selection. Some recruiters may prioritize candidates from specific educational backgrounds, while others may prioritize technical skills or work experience. As a result, there is no one-size-fits-all approach, and the effectiveness of the system can vary based on individual requirements.

Conclusion

In conclusion, the resume ranking and candidates selection chapter has successfully developed an AI-based solution that streamlines the candidate selection process for recruiters. By combining advanced text processing techniques, such as word2vec and cosine similarity, with years of experience similarity calculations, the system accurately matches job offers with the most qualified candidates. The implementation of modular programming and a flexible YAML configuration allows recruiters to easily customize the number of candidates they want to evaluate, enhancing the system's usability.

Moreover, to address the challenge that every recruiter may have specific criteria for the job offer in question we will provide a comprehensive solution, the next chapter will focus on developing a detailed dashboard that offers recruiters a wealth of insights and reporting options. The dashboard will empower recruiters to customize and fine-tune the system according to their unique preferences and priorities. It will allow them to explore candidate profiles based on various attributes, such as age, years of experience, experience level, and more.

Chapter 6

Dashboard and Reporting

Introduction

The ”Dashboard and Reporting” chapter introduces a comprehensive and user-friendly solution designed to empower recruiters with valuable tools to optimize their candidate selection process. Building upon the success of our resume ranking and candidates’ selection system, this chapter dives into the development of a dynamic dashboard that offers recruiters a wealth of visualizations, analytics, and key performance indicators.

6.1 Background and Context

We will explore three essential subsections that lay the foundation for our comprehensive dashboard and reporting solution. In the first subsection of the Background and Context section, we embark on a detailed exploration of the Extract, Transform, Load (ETL) workflow—an integral part of our comprehensive dashboard and reporting solution.

The second subsection focuses on the design of the data warehouse, a central repository that serves as the backbone of our reporting and analytics. We explore the architecture and data modeling considerations, ensuring that the data warehouse is optimized for fast querying and data retrieval. In the final subsection, we unveil the heart of our dashboard and reporting solution. We discuss the essential components of a user-friendly and insightful dashboard, emphasizing the importance of intuitive navigation, customizable filters, and interactive data visualizations.

Moreover, our dashboard and reporting solution stands as a standalone Business Intelligence (BI) project, capable of delivering comprehensive and insightful analytics. By integrating sophisticated ETL processes and a data cleaning pipeline, our system takes raw and heterogeneous data and transforms it into actionable insights. This standalone BI project offers a robust and scalable platform that empowers recruiters with the tools they need to make data-driven decisions in the candidate selection process.

6.1.1 Extract, transform, and load Workflow

We have invested significant effort in developing a robust data cleaning pipeline to optimize the reporting phase. This pipeline serves as a crucial intermediary step, where data is meticulously cleaned, standardized, and transformed into a consistent format.

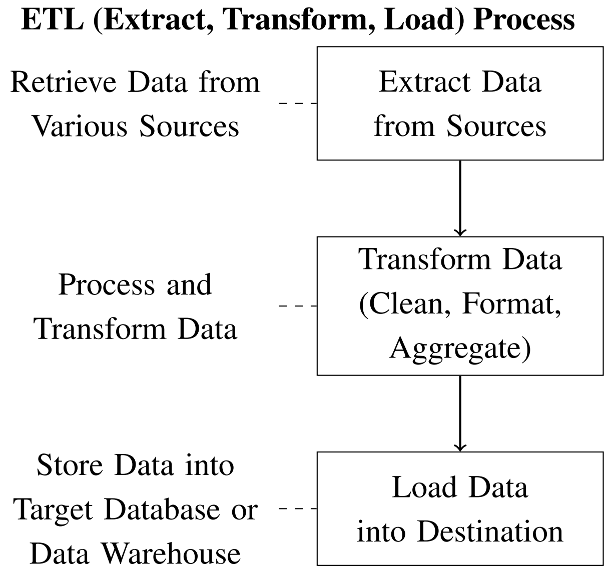

ETL (Extract, Transform, Load) Process
Retrieve Data from Various Sources
Extract Data from Sources
Process and Transform Data
Transform Data (Clean, Format, Aggregate)
Store Data into Target Database or Data Warehouse
Load Data into Destination

Figure 6.1: Illustration of the ETL (Extract, Transform, Load) Process

By efficiently handling data from various sources, such as candidate profiles and company job offers. This process consists of three essential stages: Extraction, Transformation, and Loading.

During the Extraction stage, data is collected from diverse sources, including job portals, candidate applications, and the company's internal databases. This raw data is often heterogeneous, with varying formats, structures, and quality. As a result, it is essential to carefully gather and consolidate the data to ensure its integrity and accuracy.

The Transformation stage is where the data undergoes a series of rigorous operations to clean and standardize it. The primary objective is to ensure consistency across different data sources and to remove any duplicates or errors that might impact the accuracy of the reporting.

After cleansing the data, the Loading stage involves populating the data into a designated data repository, which is called a data warehouse. This centralized repository is designed to store and organize the cleaned and transformed data in a structured manner, making it easily accessible for reporting and analysis.

6.1.2 Data Warehouse Design

In our data warehouse model, we have adopted the widely used star schema, a simple and effective data modeling technique for business intelligence and reporting purposes. The star schema consists of a central fact table surrounded by dimension tables, facilitating efficient querying and analysis of data.

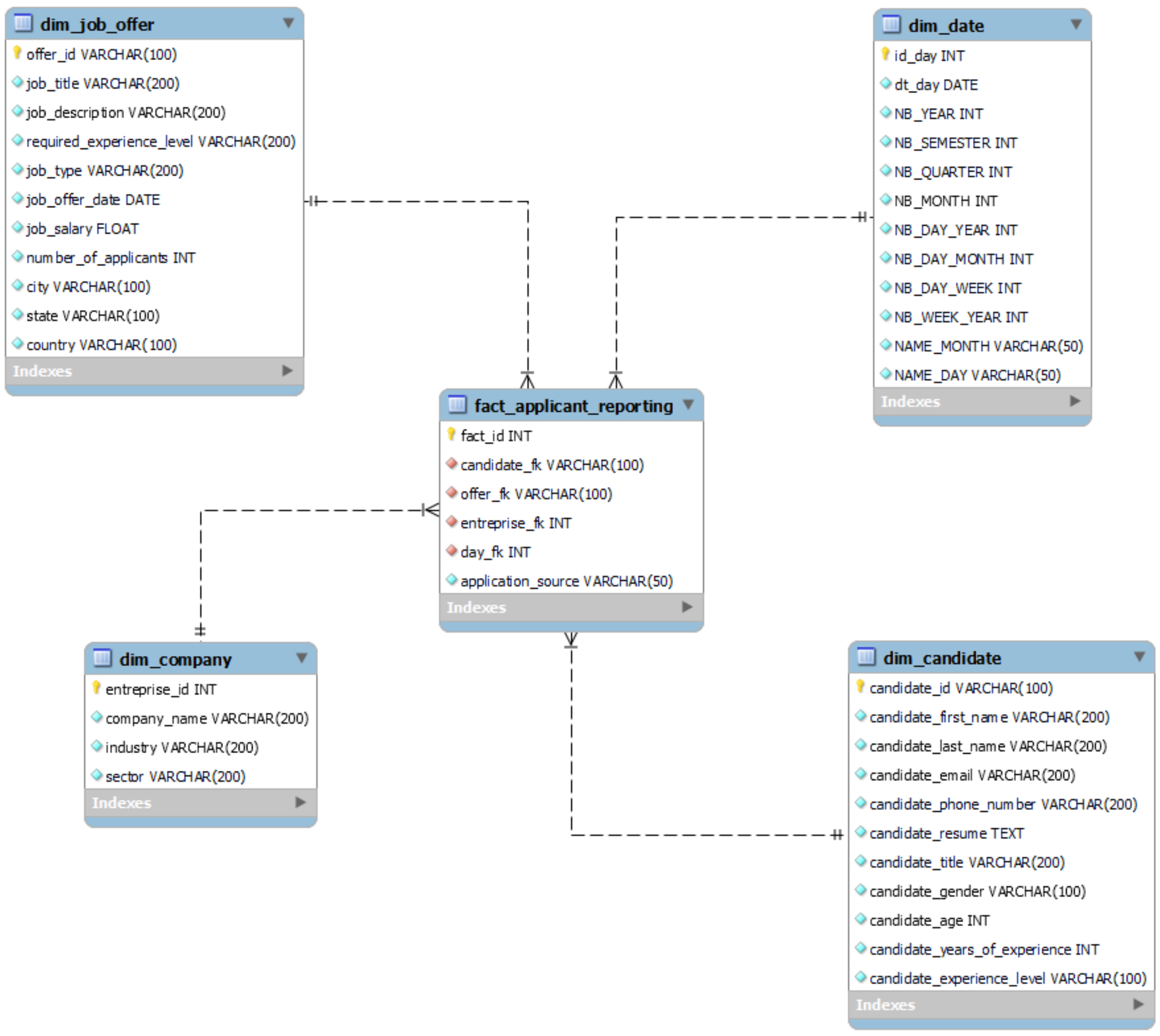


Figure 6.2: Data Warehouse Design

The fact table in our model is named ”fact_applicant_reporting”. This table serves as the core of our data warehouse and contains the quantitative metrics and measures related to the candidate selection process. It acts as a bridge between the dimension tables, capturing the numerical data and performance indicators for each candidate and job offer combination.

The first dimension table, ”dim_candidate”, contains detailed information about each candidate who applied for the job offers. It stores attributes such as candidate ID, name, age, experience level, skills, educational background, and any other relevant information that recruiters may use to evaluate candidates.

The second dimension table, ”dim_job_offer”, focuses on providing comprehensive insights into the job offers. It stores attributes such as job offer ID, job title, required years of experience, salary range, job description, and any other essential details related to the job openings.

Lastly, the third dimension table, ”dim_date_offer”, serves to capture temporal information. It includes attributes like date, month, year, and other time-related data to enable time-based analysis of candidate applications and job offer activities.

Review

The star schema design optimizes query performance, as it avoids complex joins and allows for efficient aggregations and filtering on the fact table. With the data warehouse organized in a star schema, the reporting and visualization processes become faster and more user-friendly.

6.1.3 Dashboard and Visualization

Our Dashboard and Reporting subsection will provide recruiters with a comprehensive set of Key Performance Indicators (KPIs) that offer valuable insights into the candidate selection process. These KPIs cover various aspects of the recruitment process and aim to enhance decision-making and optimize hiring strategies.

- **Number of Applicants by Job Type or Job Offer**: This KPI will display the total number of applicants received for each job type or specific job offer. It allows recruiters to gauge the level of interest in different positions and helps identify popular job opportunities.

- **Applicants' Average Years of Experience**: This KPI calculates all applicants' average years of experience. It provides recruiters with an overview of the overall experience level of candidates applying for various roles.

- **Gender Rate**: Tracking the distribution of male, female, and other gender applicants, this KPI helps assess the inclusivity and diversity of the candidate pool.

- **Experience Level Rate**: By categorizing applicants based on their experience levels (e.g., entry-level, mid-level, senior), this KPI offers valuable insights into the candidate mix and enables recruiters to target specific talent pools.

- **Applicants' Age and Years of Experience Distribution**: This KPI visualizes the age and years of experience distribution of candidates, giving recruiters a deeper understanding of the talent pool's demographics.

- **Top Applicants by Years of Experience**: Identifying the candidates with the highest years of experience for each job type or offer, this KPI helps recruiters identify potential top performers and niche experts.

Our reporting system will present these KPIs through interactive charts, graphs, and tables, allowing recruiters to analyze data with ease and precision.

6.2 Implementation and Execution

For the implementation and execution, I utilized Talend Open Studio as the primary tool for the Extract, Transform, Load (ETL) workflow. Talend's powerful data integration capabilities allowed me to efficiently extract raw data from various sources and perform essential data cleaning tasks. Within the ETL process, I incorporated numerous cleaning components, such as identifying and handling duplicate records, detecting and addressing missing values, and implementing data validation routines to ensure data quality and consistency.

For the data model, I opted for MySQL as the relational database management system. MySQL's flexibility and scalability made it well-suited for creating the star schema data warehouse model. The fact table, named ”fact$_applicant_reporting, ” servesasthecenterpiece, capturingkeymetri$

To visualize and present the wealth of data and insights, I leveraged the capabilities of Power BI, a powerful business intelligence tool. With Power BI, I designed and developed an interactive and intuitive dashboard that displays the various KPIs and metrics related to candidate selection. The dashboard provides recruiters with a user-friendly interface to explore and analyze the data, offering customizable views and filtering options.

6.3 Results and Finding

In this section, we present our comprehensive dashboard, comprising three key pages that offer valuable insights and reporting options for recruiters. The first page, ”Executive Summary,” we present a comprehensive view of the candidates and job offers information through a collection of visually impactful and insightful data visualizations. Starting from the left, the treemap provides a clear breakdown of the number of applications for each job type, enabling recruiters to quickly identify the most popular job categories.

Just below the treemap, a stacked bar chart showcases the number of applications for each job offer, sorted in descending order. This visualization offers a quick overview of the popularity of different job offers, helping recruiters prioritize their focus on high-demand positions.

Moving to the center of the page, the matrix visual presents a list of all the job offers, accompanied by the number of applications received for each offer. The visual also incorporates a user-friendly color scheme where the opacity of the color increases with the average experience level of the candidates who applied for each offer. This clever color representation allows recruiters to gauge the level of experience of applicants for each job offer at a glance.

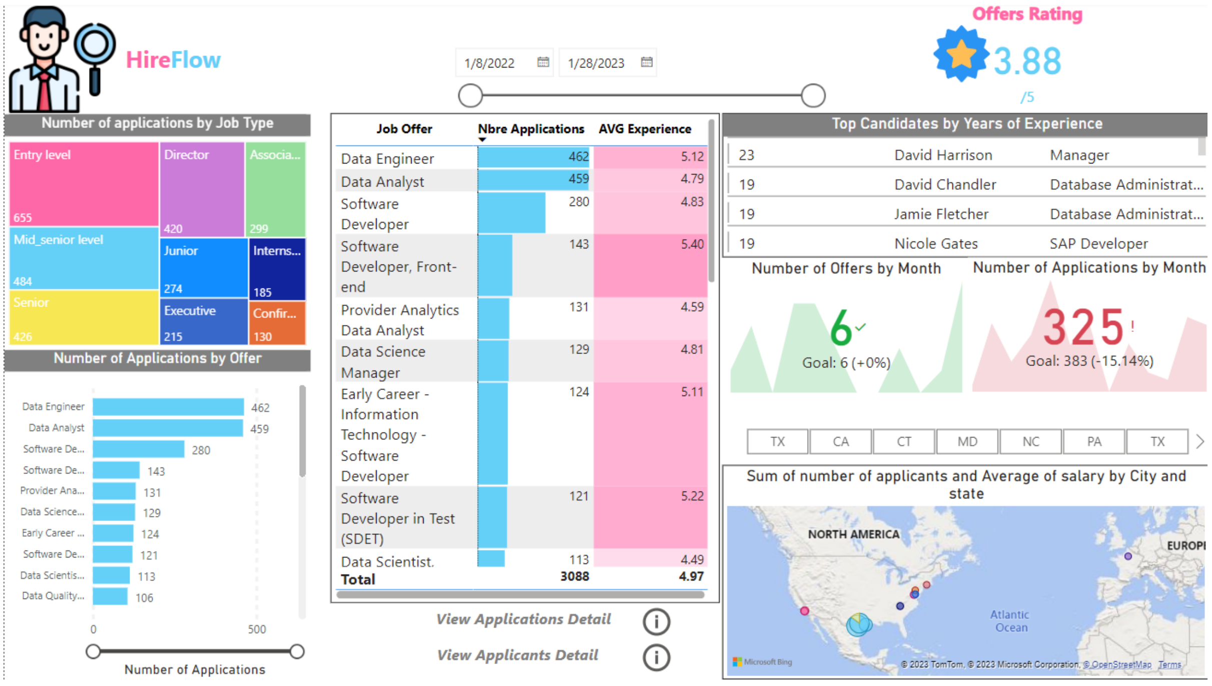
On the right side of the page, we have several insightful KPIs and cards. The simple card displays the average rating that job offers received, providing valuable information about the overall quality and attractiveness of the offers to potential candidates.

The multi-row card showcases the top experienced candidates based on their years of experience, along with their corresponding titles. This information helps recruiters identify highly qualified candidates with extensive experience relevant to specific job offers.

Additionally, we incorporated two KPIs on the right side. The first KPI illustrates the number of job offers posted by the company in the current month, compared to the previous month. This comparison aids recruiters in understanding the fluctuation in job offers over time.

The second KPI shows the number of applications received by the company in the current month, compared to the previous month. This KPI highlights trends in candidate interest and application volume, helping recruiters assess the success of their recruitment efforts.

Finally, to provide a global perspective, the map visualization displays the locations of job offers around the world. Each job offer is represented by a circle, and the size of the circle corresponds to the number of applicants for that particular offer. This visual gives recruiters a comprehensive understanding of the geographical distribution of job opportunities and candidate interest. These data visualizations are designed to be highly interactive and interconnected, allowing recruiters to explore and analyze the data dynamically. Furthermore, the inclusion of a slicer provides the recruiter with the ability to filter the data by time or period.



Job Offer	Nbre Applications	AVG Experience
Data Engineer	462	5.12
Data Analyst	459	4.79
Software Developer	280	4.83
Software Developer, Front-end	143	5.40
Provider Analytics Data Analyst	131	4.59
Data Science Manager	129	4.81
Early Career - Information Technology - Software Developer	124	5.11
Software Developer in Test (SDET)	121	5.22
Data Scientist.	113	4.49
Total	**3088**	**4.97**

Top Candidates by Years of Experience

23	David Harrison	Manager
19	David Chandler	Database Administrat...
19	Jamie Fletcher	Database Administrat...
19	Nicole Gates	SAP Developer

Figure 6.3: First Page of our Dashboard

Moving to the second page, in our dashboard, where we designed a dynamic funnel visualization that offers a clear overview of the top job boards from which applications originated. This funnel allows recruiters to easily identify the most effective job boards for attracting potential candidates, thus enabling them to focus their recruitment efforts strategically.

Additionally, we incorporated essential Key Performance Indicators (KPIs) that provide valuable insights into the current month's job offers and application trends compared to the previous month. This comparison helps recruiters understand the growth or decline in job offers and applications over time, providing a basis for analyzing the effectiveness of their recruitment strategies.

Moreover, to present a comprehensive view of the recruitment data, we integrated a line chart that displays the number of offers for each month. This chart not only showcases the historical trend in job offers but also includes a forecast for the next month, generated using relevant statistical models or forecasting techniques. The forecast empowers recruiters to anticipate future trends and prepare for potential fluctuations in job demand.

Furthermore, below the line chart, we incorporated an area chart that presents the number of applications received per month. This chart visually illustrates the volume of applications over time, helping recruiters identify patterns and seasonal trends in candidate interest.

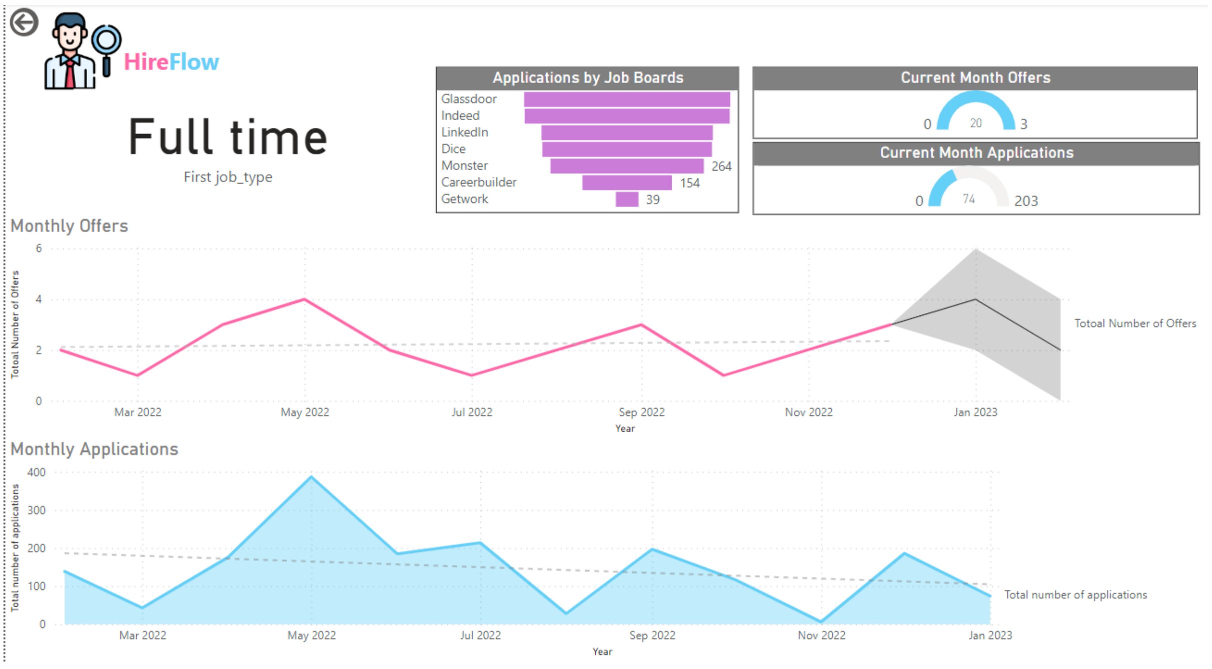


Figure 6.4: Second Page of our Dashboard

On the final page, "Applicants Details," recruiters can access other information about the candidates who have applied for specific job offers. This detailed view allows recruiters to examine candidates' Experience, gender, and previous work experiences. By using the filters the dashboard provides a comprehensive overview of each candidate, and recruiters can make well-informed decisions about which applicants to further consider for specific job offers.

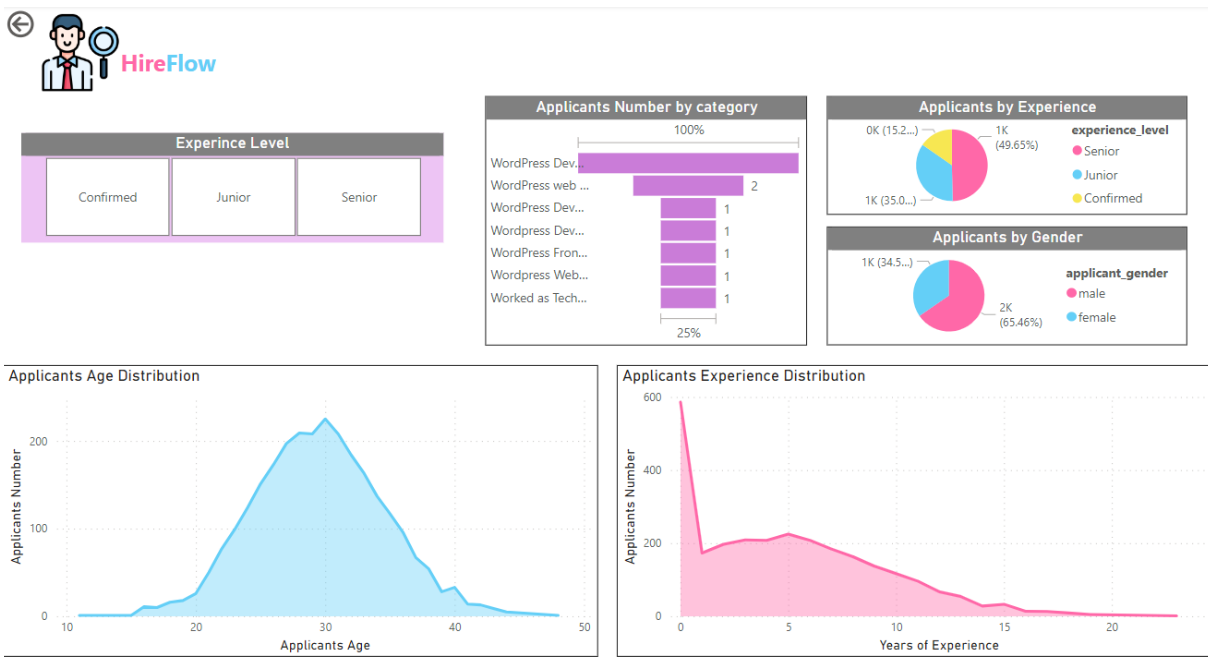


Figure 6.5: Last Page of our Dashboard

6.4 Lesson Learned and Reflection

During the implementation of the dashboard, we learned several valuable lessons that contributed to the success of the project. One of the key takeaways was the importance of choosing the right tools and technologies for the job. We found that Talend Open Studio was an excellent choice for the ETL workflow, as it provided a range of powerful cleaning components and routines to handle data extraction, transformation, and loading efficiently. Similarly, MySQL proved to be a reliable and scalable database management system for storing our data warehouse model.

Another lesson learned was the significance of user-centric design and flexibility in dashboard creation. By dividing the dashboard into three pages - ”Exec Summary,” ”Applications Details,” and ”Applicants Details” - we catered to different user needs and provided a clear and organized view of the recruitment data. Empowering recruiters with the ability to access key performance indicators (KPIs) such as applicant demographics, experience distribution, and top applicants by experience level allowed them to gain actionable insights and make informed decisions.

We also recognized the importance of data visualization in conveying complex information effectively. Utilizing Power BI for dashboarding proved to be a wise decision, as it allowed us to create interactive and visually appealing visualizations. The use of charts, graphs, and tables facilitated a better understanding and interpretation of the recruitment data.

Conclusion and Next Step

In conclusion, the integration of the AI algorithm for candidate selection, coupled with the development of an interactive dashboard, signifies a significant advancement in our data-driven solution landscape. The AI algorithm effectively identifies the most qualified candidates based on various criteria, streamlining the candidate selection process and saving valuable time for the recruiters.

However, the implementation of the interactive dashboard takes the solution to a whole new level. By providing the recruiter with a wealth of information and insights about the candidates, the dashboard empowers them to make well-informed and data-driven decisions. The ”Executive Summary” page, with its array of KPIs and data visualizations, offers a comprehensive overview of the candidates and job offers metrics, enabling the recruiter to quickly grasp key trends and patterns.

Moreover, the dashboard's interactivity and interconnectedness allow recruiters to delve deeper into the data, exploring candidate profiles based on specific attributes such as age, years of experience, and experience level.

In the next chapter, we will shift our focus to the critical evaluation phase, where we will rigorously assess the performance and effectiveness of our recommender systems. This evaluation step is of utmost importance as it allows us to validate the accuracy, reliability, and overall utility of our recommendation models. By conducting a comprehensive evaluation, we can gain valuable insights into how well our systems meet the specific objectives and requirements set at the beginning of our project.

Chapter 7

Evaluation

Introduction

In this chapter, we circle back to the core of our recommender systems—the algorithms that were carefully designed and implemented. Here, we embark on a thorough evaluation journey, seeking to assess the performance, accuracy, and overall effectiveness of our recommendation models.

7.1 Background and Context

The challenge with recommender systems lies in their elusive nature, as they encompass both scientific and artistic elements, making it challenging to gauge their true efficacy. Evaluating their performance becomes a subjective endeavour, given the aesthetic appeal of their outcomes, and the difficulty in ascertaining a user's perception of a recommendation—particularly in offline algorithm development.

Apart from accuracy, there exist numerous metrics[17] to evaluate recommender systems, and these measurements can often conflict with one another. People have devised diverse methodologies to gauge the system's quality, leading to varied and sometimes contradictory assessments.

7.1.1 Testing Methodologies

These testing techniques will play a crucial role in evaluating the performance and effectiveness of our recommender systems. By quantifying the accuracy of our models through rigorous testing, we can ensure that our recommendations are reliable and accurate.

Train Test Split

In the train-test split, we partition our dataset into two separate subsets: the training set and the testing set. The training set is used to train our recommender models, while the testing set is employed to evaluate the model's performance.

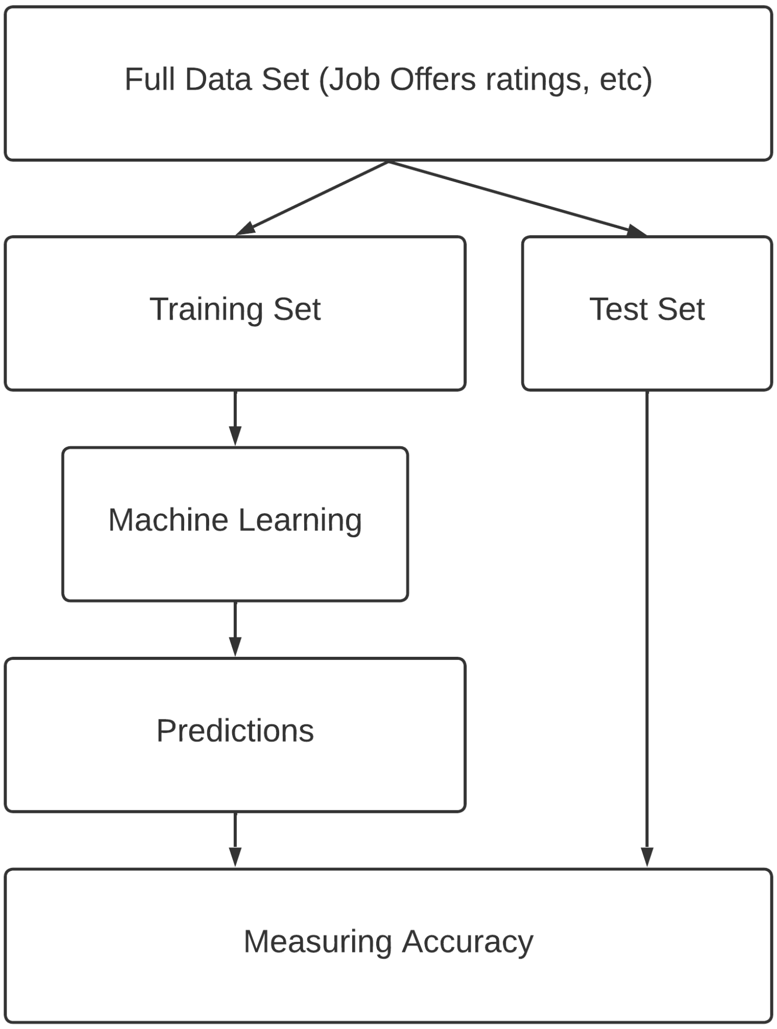


Figure 7.1: Train Test Split

K-fold Cross Validation

We can enhance our evaluation process by utilizing k-fold cross-validation[18], a more sophisticated technique than a simple train/test split. With k-fold cross-validation, we divide our dataset into k randomly assigned subsets or "folds." Each fold is used as a separate training set to train our recommender system, and then its accuracy is measured against the remaining data in the test set. This process is repeated k times, with each fold serving as both a training and test set. By averaging the accuracy scores from these iterations, we obtain a more reliable and robust estimate of our recommender system's performance in predicting user ratings. K-fold cross-validation allows us to assess how well our models generalize to unseen data, making it a valuable tool for refining and validating our recommender systems.

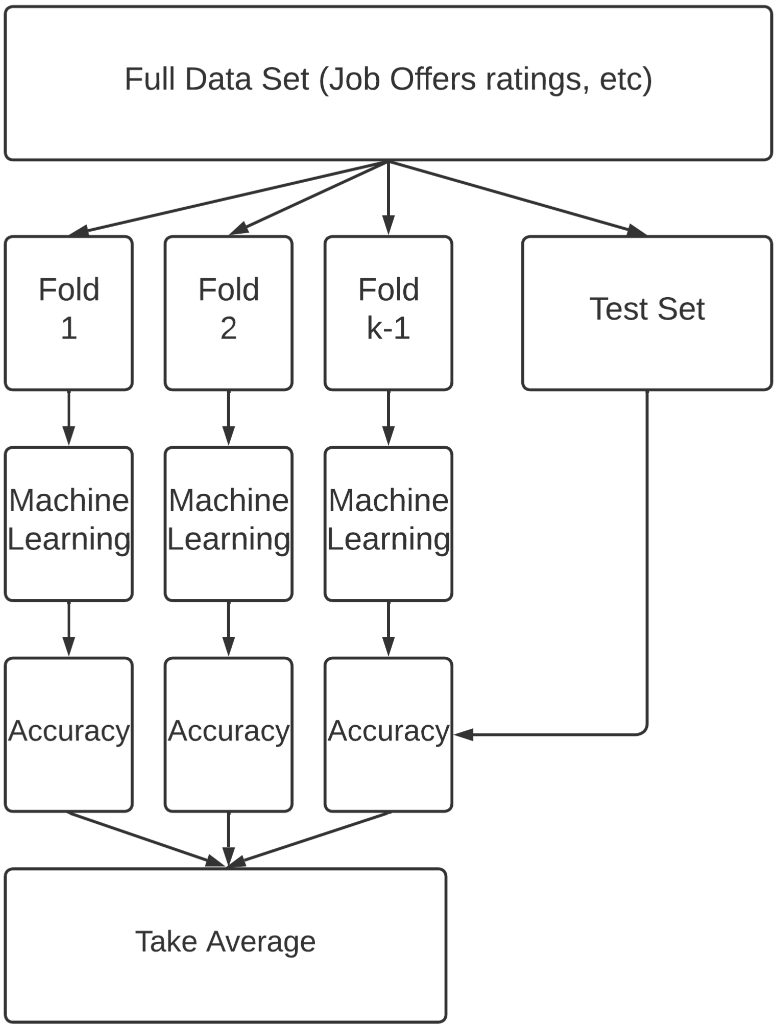


Figure 7.2: K-fold Cross Validation

K-fold cross-validation demands more computing power, but its advantage lies in mitigating the risk of ”overfitting” to a single training set. By using multiple randomly assigned training sets and averaging their performance, K-fold cross-validation provides a more robust evaluation of the recommender system's generalization ability.

7.1.2 Accuracy Measures

After splitting our data, we can measure the accuracy of our recommender systems using two key metrics: the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE).

Mean Absolute Error (MAE)

The most straightforward metric for evaluating recommender systems is the Mean Absolute Error (MAE), which measures the average absolute difference between the predicted ratings and the actual ratings given by users. The mathematical equation for MAE is as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

In this equation, n presents the number of user ratings, $\hat{y}_i$ presents the predicted rating by our machine learning model, and y_i presents the real job offer rating.

Root Mean Square Error (RMSE)

Another commonly used metric for measuring accuracy is the Root Mean Square Error (RMSE), which calculates the square root of the average of the squared differences between the predicted ratings and the actual ratings given by users. The mathematical equation for RMSE is as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

In this equation, n presents the number of user ratings, $\hat{y}_i$ presents the predicted rating by our machine learning model, and y_i presents the real job offer rating.

MAE and RMSE both assess the accuracy of predictions, but RMSE is more sensitive to large errors due to the squared term, while MAE treats all errors equally. The choice between MAE and RMSE depends on the specific use case and the level of sensitivity to outliers in the evaluation process. Accuracy, in the context of recommender systems, may not be the most relevant metric for assessing their effectiveness. The ultimate goal of a recommender system is not merely to predict ratings for items a user has already rated but to provide valuable suggestions for new items they may enjoy. For that, we will implement the Hit Rate measures.

7.1.3 Hit Rate Measures

Accuracy, in the context of recommender systems, may not be the most relevant metric for assessing their effectiveness. The ultimate goal of a recommender system is not merely to predict ratings for items a user has already rated but to provide valuable suggestions for new items they may enjoy. For that, we will implement the Hit Rate measures.

Hit Rate

When measuring the Hit Rate, we begin by generating top-N recommendations for each user in the test set. If any of the recommended items from this list are among the items that the user has already rated, we consider it a ”hit.” This implies that the recommender system successfully suggested something the user had already shown interest in, indicating a successful recommendation.

Cumulative Hit Rate

Cumulative hit rank, while it may sound complex, is a straightforward concept. It involves discarding hits from our recommendations if the predicted rating falls below a certain threshold. The rationale behind this approach is simple: we shouldn't count recommendations for items that we believe the user won't find enjoyable or relevant.

Rating Hit Rate

Rating Hit Rate (rHR) provides a breakdown of hit rate based on predicted rating scores, offering insights into the distribution of the algorithm's perception of recommended offers that users genuinely liked. It allows us to assess how well our system recommends job offers that align with users' preferences and interests. The focus is on delivering high rHR values, indicating that the recommendations are accurately matching user preferences.

Average Reciprocal Hit Rate

Average Reciprocal Hit Rate (ARHR) is a modified version of the hit rate metric, taking into account the position of the hits in the top-N list. It gives higher credit for successful recommendations that appear in the top positions, emphasizing the items users are most likely to consider. This user-centric metric aligns with the fact that users often concentrate their attention on the beginning of the recommendation lists. Below is the fancy mathematical equation of the Average Reciprocal Hit Rate:

$$\text{ARHR} = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{\text{rank}_k}$$

In this equation, N is the number of recommended items, and rank_k is the position of the k-th hit in the list.

7.1.4 Coverage

Recommender systems' evaluation goes beyond just accuracy; other crucial aspects can be measured based on their relevance. One such aspect is **coverage**, which refers to the percentage of potential recommendations that our system can offer. It highlights the system's ability to provide a comprehensive set of recommendations across a diverse range of items:

% of ¡job seeker, offer¿ pairs that can be predicted.

7.1.5 Diversity

Diversity, another key metric, gauges the range of job offers presented by our recommender system to job seekers. However, it's crucial to consider that diversity doesn't always indicate effectiveness. Recommending completely random job offers can achieve high diversity but may not lead to meaningful results. Striking a balance between diversity and relevance is essential for an effective recommender system:

Diversity = 1 - S

S: Average similarity between recommender job offers.

7.1.6 Novelty

Novelty is a metric that reflects the popularity of the recommended job offers. It gauges how unique or distinct the suggested offers are compared to commonly favored options.

7.2 Implementation and Execution

In the Implementation and Execution phase, we utilized the convenient implementation provided by SurpriseLib for calculating the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics. These metrics were essential to evaluate the accuracy of our recommender systems. Additionally, we developed custom implementations for other metrics like Hit Rate, Average Reciprocal Hit Rate (ARHR), Rating Hit Rate (rHR), Coverage, and Diversity. These custom implementations allowed us to assess various aspects of our recommender systems, providing valuable insights into their performance and effectiveness. The combination of both pre-existing and custom-built implementations enabled a comprehensive evaluation of our models to ensure their relevance and usefulness in real-world scenarios.

7.3 Results and Finding

In the Results and Findings section, we present a comprehensive table showcasing the results of various evaluation metrics applied to our developed recommender systems. The table includes the calculated values for metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Hit Rate, Average Reciprocal Hit Rate (ARHR), Rating Hit Rate (rHR), Coverage, and Diversity. Each row of the table corresponds to a specific recommender system, and the columns represent the different metrics. This tabular representation allows us to compare the performance of each model across different evaluation measures, providing a clear understanding of their strengths and weaknesses.

Table 7.1: Evaluation Metrics for Different Recommendation Algorithms

Algorithm	RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
ContentKNN	0.4469	0.3831	0.1500	0.1500	0.0609	0.9900	0.0103	135.1232
User KNN	0.4445	0.3830	0.1100	0.1100	0.0383	0.9900	0.9903	121.3961
Item KNN	0.4387	0.3778	0.1100	0.1100	0.0370	0.9900	0.9903	136.7994
SVD	0.4502	0.3853	0.1600	0.1600	0.0597	0.9900	0.9903	133.9450
SVD++	0.4467	0.3829	0.1300	0.1300	0.0575	0.9900	0.9903	130.0998
RBM	1.1878	1.1036	0.2000	0.2000	0.0785	0.9900	0.0104	49.4644

RMSE: The Root Mean Squared Error (RMSE) serves as a measure of accuracy, with lower values indicating better predictive performance.

MAE: The Mean Absolute Error (MAE) is another accuracy metric, and lower values signify improved precision in the predictions.

HR: The Hit Rate measures how frequently we can successfully recommend a left-out rating, and a higher value indicates better performance.

cHR: The Cumulative Hit Rate (cHR) is a variation of the Hit Rate that focuses on ratings above a specific threshold, and a higher cHR value indicates improved performance.

ARHR: The Average Reciprocal Hit Rank (ARHR) is a Hit Rate metric that considers the ranking of recommendations, and higher ARHR values indicate better performance.

Coverage: Coverage is the ratio of users for whom recommendations above a certain threshold exist. A higher coverage value indicates better coverage of the user base with recommendations.

Diversity: Diversity is measured as 1 minus the average similarity score (S) between every possible pair of recommendations for a given user. A higher diversity score indicates a more diverse set of recommendations.

Novelty: Novelty is quantified as the average popularity rank of the recommended job offers. A higher value means more novel.

7.4 Lesson Learned and Reflection

During the evaluation of recommender engines, I gained valuable insights into various evaluation metrics and successfully implemented several of them. Covering a range of offline metrics allowed us to assess the performance and accuracy of our models based on historical data. However, I came to realize that while offline metrics provide useful indicators, they may not fully capture the real-world performance and impact of our recommender system. The results of real-world scenarios and online A/B tests are the ultimate evaluation that truly matters. These tests involve exposing the recommender system to real users in real-time, allowing us to measure its effectiveness and user satisfaction in a dynamic and dynamic environment. Such evaluations provide critical feedback on how well the system performs in practical situations, taking into account various user interactions and feedback. By combining both offline and online evaluations, we can achieve a more comprehensive understanding of our recommender system's strengths and areas for improvement, ultimately leading to a more refined and effective solution.

Conclusion and Next Step

In conclusion, this chapter delved into the evaluation of our recommender systems using a variety of metrics that provide insight into their performance and effectiveness. Through the application of methodologies like train/test split and k-fold cross-validation, we gained a deeper understanding of how well our systems are performing. The results showcased the strengths and weaknesses of different algorithms, shedding light on their capabilities and areas for improvement. While offline metrics are valuable indicators, we acknowledge that the true test lies in real-world scenarios and online A/B tests. As the next step, we will embark on the journey of scaling up our system to handle larger datasets, ensuring its robustness and efficiency in addressing the challenges of real-world applications.

Chapter 8

Scaling Up the System

Introduction

Until now, we have utilized a small rating dataset for job offers to showcase various recommender systems. While this has served educational purposes, building a recommender system for a real company would involve dealing with significantly larger datasets, far beyond what a single desktop PC can handle. In this section, we delve into the process of scaling up our recommender systems to handle real ”big data” scenarios. We explore solutions that allow us to train recommender systems using clusters or even in the cloud, providing the necessary computational power and resources to handle massive datasets and ensure efficient and scalable performance.

8.1 Background and Context

One of the most effective ways to handle large datasets is through the use of Apache Spark. As a distributed computing framework, Spark enables us to process massive amounts of data efficiently and in parallel across multiple nodes in a cluster.

8.1.1 Apache Spark Architecture

Spark’s significance lies in its exceptional efficiency when it comes to distributing the processing of enormous datasets across a cluster in a robust and reliable manner. Architecturally[9], it is structured to function as depicted in the image below.

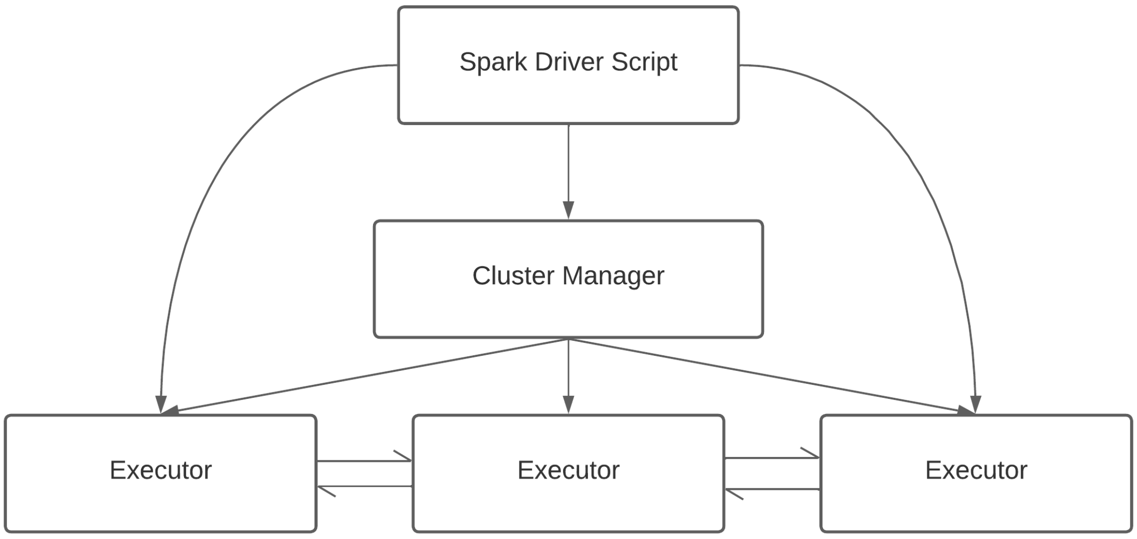


Figure 8.1: Apache Spark Architecture

To leverage the power of Apache Spark, we begin by creating a driver script in Python, Scala, or Java that outlines how we want to process our data using Spark's APIs. Once we launch this driver script, typically from the master node of our cluster, it communicates with the cluster manager to secure the required resources from the cluster. Depending on our setup, the cluster manager could be Hadoop's YARN if we are running Spark on a Hadoop cluster or Spark's own cluster manager. The cluster manager then distributes executor processes across the cluster to handle the actual data processing. Spark efficiently divides and processes the data, optimizing memory usage and performing computations on the nodes where the data is stored. To organize the processing most effectively, Spark employs a directed acyclic graph. Additionally, Spark ensures seamless communication among the various components, allowing them to respond to situations like machine failures within the cluster.

8.1.2 Apache Software

From a software developer's standpoint, Spark comprises a core responsible for managing all the work distribution we previously discussed, complemented by additional libraries that are commonly utilized.

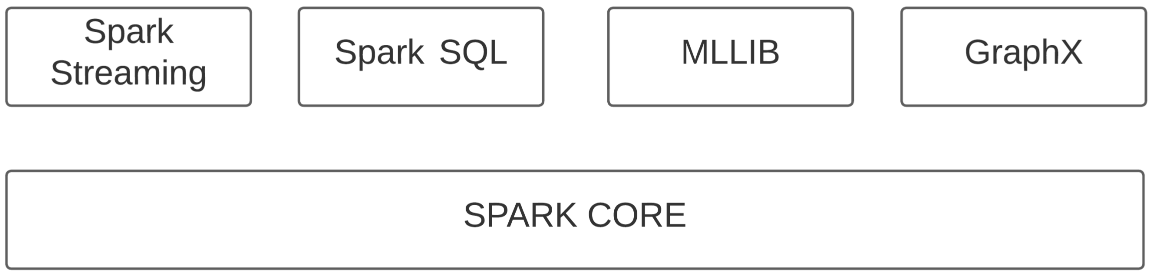


Figure 8.2: Spark Software Architecture

For instance, Spark SQL introduces DataSets, allowing us to work with Spark in a similar manner to SQL databases. DataSets have become the preferred method of utilizing Spark due to their promising potential. Spark Streaming facilitates real-time data ingestion and processing, with Structured Streaming enabling us to treat real-time data as SQL data. GraphX serves the purpose of analyzing and processing data structured in graph format, such as social networks. However, our main focus lies on MLLib, Spark's machine learning library, which plays a vital role in generating recommendations from large-scale data sets. Its user-friendly classes make recommendation generation a straightforward task, enabling us to build robust and efficient recommender systems.

8.2 Implementation and Execution

In the implementation and execution of our recommender systems on Apache Spark, we employed PySpark, a Python implementation of Spark, to leverage the distributed computing capabilities of Spark. PySpark provided us with a user-friendly and flexible environment to work with large-scale data and develop sophisticated recommendation algorithms.

For the recommendation models, we chose the Alternating Least Squares (ALS) algorithm, a widely used collaborative filtering method in Spark's MLlib library. ALS attempts to estimate user-item ratings based on the interactions between users and items in the data. It is a matrix factorization technique, similar to the SVD method we discussed in the Recommender Engine chapter. In fact, in that chapter, we referred to it as SVD++, which is an extension of SVD with additional optimization algorithms for the cost function.

The ALS algorithm in PySpark efficiently handles the task of matrix factorization, breaking down the user-item interaction matrix into two lower-rank matrices, one for users and the other for items. By iteratively optimizing these matrices, ALS predicts the missing ratings for users and items, ultimately providing us with personalized recommendations for each user. The decision to use the ALS algorithm was driven by its proven effectiveness in generating accurate and personalized recommendations in large-scale scenarios. PySpark's seamless integration with ALS enabled us to focus on fine-tuning and optimizing the algorithm to achieve the best possible results for our recommendation system.

In summary, the combination of PySpark's capabilities and the ALS algorithm proved to be a winning formula for building robust and scalable recommender systems. By leveraging PySpark and utilizing the ALS algorithm, we could efficiently process massive data sets and deliver accurate, personalized recommendations to our users, enhancing their experience and satisfaction with the recommendation platform.

8.3 Results and Finding

In order to see the results of our Spark-based recommender system, we conducted a test on the same user we used in the recommender engine chapter – a data scientist seeking job opportunities. Running the Spark implementation on our local machine allowed us to analyze the recommendations and assess their relevance and accuracy for the user.

The table below showcases the recommended job offers for the data scientist user.

Table 8.1: ATL Results

Recommendations	Predicted Rating
Data Scientist	4.548089059224887
Technical Data Consultant	4.485303410377958
Data Scientist, Marketplace Economics	4.475899130212679
Data Engineer	4.472740417044908
Data Scientist, Acorn AI Labs	4.4677359860451835
Senior Data Scientist, Real World EvidenceNY	4.464480172492173
Data Scientist, Voter Data	4.462619662837714
Data Scientist, Decisions	4.4624205682078815
AI Engineering: Data Scientist/Machine Learning Engineer	4.456574612260545
Quantitative Researcher	4.4553675164905435

As evident from the results, the Spark-based recommender system demonstrates striking similarities to the SVD++ algorithm tested in the recommender engine chapter. This outcome is not surprising, as both implementations utilize the same algorithm and process the same dataset on the same hardware. The consistency in results validates the accuracy and reliability of our Spark-based approach in generating recommendations for job offers. However, it also highlights the importance of thorough testing and evaluation to ensure that our system performs consistently across different implementations and scenarios.

8.4 Lesson Learned and Reflection

Throughout the process of scaling up our recommender system using Apache Spark, we gained valuable insights and learned essential lessons to improve the efficiency and effectiveness of our algorithms. One of the key takeaways was the significance of distributing the processing of massive datasets across a cluster. Spark's architecture allowed us to efficiently manage the distribution of work and reliably process extensive data on a scalable platform. By leveraging Spark's machine learning library, MLLib, we simplified the process of generating recommendations from large datasets, enhancing the system's performance and accuracy.

Moreover, we recognized the importance of utilizing Spark's various libraries, such as Spark SQL, Spark Streaming, and GraphX, to work with different data structures and handle real-time data ingestion and processing. The versatility of these libraries provided us with the flexibility to adapt to diverse data scenarios and cater to specific use cases.

In the reflection, we acknowledge that Spark's power comes with the trade-off of increased computing resources and infrastructure requirements. Deploying Spark on a cluster, especially on cloud platforms, offers significant advantages in terms of scalability, but it can also be cost-prohibitive. As software developers, we must carefully consider the balance between performance and budget constraints when deciding on the deployment strategy.

Conclusion

We have successfully achieved significant milestones in developing a scalable and effective recommender system using Apache Spark and PySpark. The ALS algorithm, combined with Spark's distributed computing capabilities, enabled us to handle large-scale data sets efficiently and generate accurate recommendations for our users. However, as we look ahead, we recognize that there are even more advanced deployment options available to us, such as AMAZON DDSTNE or AWS SageMaker.

Both AMAZON DDSTNE and AWS SageMaker offer cutting-edge machine learning capabilities and allow for the deployment of recommender systems at scale. These platforms provide robust infrastructures for training and deploying complex recommendation algorithms on cloud-based environments. Utilizing these platforms could further enhance the performance and scalability of our recommender system.

However, it's essential to consider the cost implications of deploying our algorithm using these advanced platforms. AMAZON DDSTNE and AWS SageMaker can be expensive, especially when dealing with massive data sets and high computational requirements. Therefore, before proceeding with such deployment options, careful cost-benefit analysis and resource planning are necessary to ensure cost-effectiveness and efficient utilization of resources.

General Conclusion

In the journey of this thesis, we embarked on a quest to redefine the recruitment landscape by harnessing the potential of data science, artificial intelligence, and natural language processing. Through rigorous exploration, analysis, and innovation, we aimed to bridge the gap between job seekers and recruiters, streamlining the job search experience and enhancing the quality of candidate selection.

As we draw the curtains on this phase of our academic journey, it's crucial to emphasize that our journey does not end here. In fact, it marks the beginning of an ongoing commitment to the continuous enhancement of our recruitment system. While we've laid the foundation for an efficient and data-driven recruitment ecosystem, we recognize that maintaining the accuracy and relevance of our AI systems, especially the recommender engines, presents an enduring challenge.

One of the cornerstones of our commitment to improvement is the incorporation of user feedback and online metrics into our system's development. Feedback from job seekers, recruiters, and other stakeholders is invaluable in identifying areas for refinement. By actively seeking and analyzing user input, we can fine-tune our recommendation algorithms, ensuring that the job recommendations provided align even more closely with individual preferences and qualifications.

As we look ahead, one of the most formidable challenges we anticipate is what we colloquially refer to as the ”cold start” scenario. In this scenario, we encounter a situation where we have limited or no historical data about a user. This is particularly challenging for recommender systems, as they rely on user behavior and preferences to generate personalized recommendations.

To address the ”cold start” challenge, we must employ innovative strategies. One approach is to leverage contextual information, such as user location or temporary data like cookies. By examining a user's region or previous interactions with the platform, we can make initial recommendations that are contextually relevant.

Moreover, we can employ hybrid recommender systems that combine collaborative filtering and content-based recommendation techniques.

In conclusion, our vision for the future is one where data-driven insights and recruitment excellence converge to create a more inclusive, efficient, and effective job market. The path forward is illuminated by the possibilities that lie ahead, and we are eager to explore and shape the future of recruitment together.

Bibliography

[1] Charu C Aggarwal and Charu C Aggarwal. Neighborhood-based collaborative filtering. *Recommender Systems: The Textbook*, pages 29–70, 2016.

[2] Ana Azevedo and Manuel Filipe Santos. Kdd, semma and crisp-dm: a parallel overview. *IADS-DM*, 2008.

[3] Rasmus Bro and Age K Smilde. Principal component analysis. *Analytical methods*, 6(9):2812–2831, 2014.

[4] Frank KY Chan and James YL Thong. Acceptance of agile methodologies: A critical review and conceptual framework. *Decision support systems*, 46(4):803–814, 2009.

[5] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46, 2010.

[6] Usama M Fayyad, David Haussler, and Paul E Stolorz. Kdd for science data analysis: Issues and examples. In *KDD*, pages 50–56, 1996.

[7] Andreas Hoecker and Vakhtang Kartvelishvili. Svd approach to data unfolding. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 372(3):469–481, 1996.

[8] Rajeev Kumar, BK Verma, and Shyam Sunder Rastogi. Social popularity based svd++ recommender system. *International Journal of Computer Applications*, 87(14), 2014.

[9] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. *The journal of machine learning research*, 17(1):1235–1241, 2016.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[12] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*, pages 325–341. Springer, 2007.

[13] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, volume 4, page 1, 2012.

[14] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

[15] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798, 2007.

[16] Christoph Schröer, Felix Kruse, and Jorge Marx Gómez. A systematic literature review on applying crisp-dm process model. *Procedia Computer Science*, 181:526–534, 2021.

[17] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. *Recommender systems handbook*, pages 257–297, 2011.

[18] Tzu-Tsung Wong and Po-Yang Yeh. Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1586–1594, 2019.

[19] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

2022 / 2023

ESPRIT SCHOOL OF ENGINEERING

www.esprit.tn - E-mail : contact@esprit.tn

Siége Social : 18 rue de l'Usine - Charguia II - 2035 - Tél. : +216 71 941 541 - Fax. : +216 71 941 889

Annexe : 1-2 rue André Ampère - 2083 - Pôle Technologique - El Ghazala - Tél +216 70 250 000 - Fax +216 70 685454